Journal of Mathematics in Industry
a SpringerOpen Journal

RESEARCH Open Access

# A holistic fast and parallel approach for accurate transient simulations of analog circuits

Janos Benk*, Georg Denk and Konrad Waldherr

*Correspondence:
benkjanos@googlemail.com
Infineon Technologies, Am Campeon 1-12, Neubiberg, 85579, Germany

**Abstract**

The accurate analog simulation of critical circuit parts is a key task in the R&D process of integrated circuits. With the increasing complexity of integrated circuits it is becoming cumulatively challenging to simulate in the analog domain and within reasonable simulation time. Previous speedup approaches of the SPICE (Simulation Program with Integrated Circuit Emphasis) analog circuit simulator included either solver improvements and speedup or model order reduction of the semiconductor devices.

In this paper we present a comprehensive approach to significantly speedup a SPICE-based analog circuit simulator while keeping the single-rate characteristic of time domain simulations. The novelty of our approach consists in the combination and extension of existing approaches in a unique way, enabling fast transient SPICE-level simulations. The main component of our approach is the circuit partitioner that combines relevant aspects from circuit theory and linear algebra in a unifying way. This enables the construction of an efficient and parallel BBD (bordered block diagonal) solver. Furthermore, this BBD structure allows for intrinsic model order reduction of the partitions during the Newton iteration, transforming the Newton method to a Quasi-Newton method.

For mid-sized and large-sized circuits our BBD approach leads to significant sequential and parallel accelerations of transient simulations. Additional speedup can be gained from our block-bypass strategies exploiting the latency in the partitioned circuit. Altogether our approach leads to a speedup of up to two orders of magnitude compared to the state-of-the-art KLU solver while maintaining SPICE-level accuracy.

**Keywords:** SPICE; analog circuit simulation; transient analysis; circuit partitioning; differential-algebraic equations; Newton method; parallel numerics; numerical linear algebra; bordered block diagonal

## 1 Introduction

The introduction of the first SPICE [1] simulator revolutionized the design of electrical circuits. The ability to simulate the circuit in various conditions and scenarios, before the circuit is actually built, is crucial for fast and cheap circuit development.

State-of-the-art commercial [2, 3] and open-source [4, 5] SPICE simulators offer numerous analyses in different physical domains and for various circuit types, helping the engineers to analyze the behavior of the circuit under different conditions. One of the most basic and workhorse-like analyses of SPICE simulators within a commercial R&D environment is the transient analysis which computes the behavior of a circuit in the time domain up to a specified time point. Although one might consider the transient analysis as consecutive operating point analyses in the time domain, we consider the transient analysis as a separate analysis. Anyway the presented methods exploit some of the particular aspects of the transient analysis which could not be applied within an operating point analysis.

The computational overhead of a transient analysis is increasing with the circuit size and with the number of time steps required during the discrete time integration. With constantly growing circuit complexity the transient simulation poses a significant bottleneck in the circuit design and verification pipeline. Hence, we focus in this paper on a holistic approach to improve the performance of the transient analysis in a general SPICE simulator, while maintaining the same level of accuracy. The proposed methods from this paper were implemented within the frame of the analog in-house circuit simulator of Infineon called TITAN [6–8]. However, our approaches are of general character and can be exported not only to other analog simulators but also to more general nonlinear transient problems.

SPICE simulators like TITAN use the modified nodal analysis (MNA) to build up the mathematical representation of the circuit. This formulation is mainly based on Kirchhoff's current law stating that the balance of all incoming and outgoing currents in a node must sum up to zero. The additive contribution of each individual device is specified by its characteristic equation which might have static and/or dynamic contents. The unknowns of the mathematical system essentially represent the node voltages; however, for each voltage-defining element such as voltage sources, an additional variable has to be introduced which represents the contribution of its branch current in the Kirchhoff's current law formulation. Altogether this approach leads to a system of differential-algebraic equations of the following general form

$$\frac{d}{dt}q\bigl(x(t)\bigr) + f\bigl(x(t)\bigr) + s(t) = 0. \tag{1}$$

Here $q,f \in \mathbb{R}^n$ are the charge vector and the resistive current vectors, whereas $s \in \mathbb{R}^n$ is the stimulus vector. The vector $x(t) = (v(t), i(t)) \in \mathbb{R}^n$ represents the unknowns of the system: the node voltages $v$ and the MNA branch currents $i$, respectively. As a starting point for the transient analysis the values of $x(0) = (v(0), i(0))$ are given or computed by an operating point analysis. Starting at $t = 0$ the goal of the transient analysis is to compute the unknown values for $t \in [0, T]$, where $T$ represents the stop time of the simulation. For more details on the circuit modeling we refer to [9].

The DAE system (1) is discretized for time points $t_k \in (0, T]$, $k = 1, \ldots, M$. Considering the backward Euler formula as a simple representative of an implicit integration scheme, the discrete form of (1) at time point $t_k$ is

$$\frac{q(x(t_k)) - q(x(t_{k-1}))}{h_k} + f\bigl(x(t_k)\bigr) + s(t_k) = 0, \tag{2}$$

with the time step $h_k = t_k - t_{k-1}$.

For each time point $t_k$ a system of nonlinear equations, derived from the discretized DAE system (2), needs to be solved with the unknown vector $x_k = x(t_k)$:

$$\frac{1}{h_k} \cdot q(x_k) + f(x_k) - \frac{1}{h_k} \cdot q(x_{k-1}) + s(t_k) = 0. \tag{3}$$

More generally, an implicit integration method takes the form

$$\frac{d}{dt} q\big(x(t)\big)\bigg|_{t=t_k} = \alpha_k q(x_k) + \beta_k,$$

with the integration coefficient $\alpha_k \in \mathbb{R}$ and the history information $\beta_k \in \mathbb{R}^n$ which are both constant for the current time point. Applied to the differential-algebraic system (1), the nonlinear system at time point $t_k$ for the unknown $x_k$ reads

$$\alpha_k q(x_k) + \beta_k + f(x_k) + s(t_k) = 0. \tag{4}$$

Applying the Newton-Raphson method transforms this nonlinear system into a sequence of linear problems: The linearization of (4) at the current value of $x_k$ specifies the Newton correction $\Delta x_k$ as the solution of the linear system

$$\left(\alpha_k \frac{dq(x_k)}{dx_k} + \frac{df(x_k)}{dx_k}\right) \Delta x_k = -\big(\alpha_k q(x_k) + \beta_k + f(x_k) + s(t_k)\big). \tag{5}$$

We denote by $C(x_k)$ and $G(x_k)$ the Jacobian matrices of the charge and of the resistive current vectors. Then $A(x_k) = \alpha_k \cdot C(x_k) + G(x_k)$ represents the combined system matrix. The residual on the right-hand side of (5) is referred to as $r(x_k)$, so that the Newton correction $\Delta x_k$ is the solution of the linear system $A(x_k)\Delta x_k = -r(x_k)$.

The linear system (5) is the foundation of the Newton method which is required for each time step $k$. The Newton loop is the most inner loop (see Figure 1) within the transient analysis and represents the largest portion of the computational task within a transient simulation.

In order to speed up the transient simulation while maintaining the SPICE-level accuracy, the algorithm on Figure 1 should be significantly improved, while keeping the single-rate and implicitly coupled characteristic of the algorithm.

Abandoning the single-rate principle or the implicit characteristic of the algorithm on Figure 1 results in a fast-SPICE algorithm [10, 11] that especially for large circuit have the potential to be 10-100 times faster than their SPICE counterparts but are considerably inaccurate and with default settings, without circuit specific settings, there is a high probability of producing wrong results. However, for large circuits and long transient simulations, the only feasible analog simulation is through such fast-SPICE algorithms.

Partitioning the circuit is a fundamental element in most of the fast-SPICE acceleration techniques. In one of the fast-SPICE approaches [12, 13] the partitions are solved implicitly but are coupled explicitly in a multi-rate manner. In this case the static partitioner [12] of the circuit must capture each feedback loop within the same partition [13] and decouple the circuit along weak capacitive connections. A similar approach is presented in [14], where the feedback loops are also included in the same partition, but the

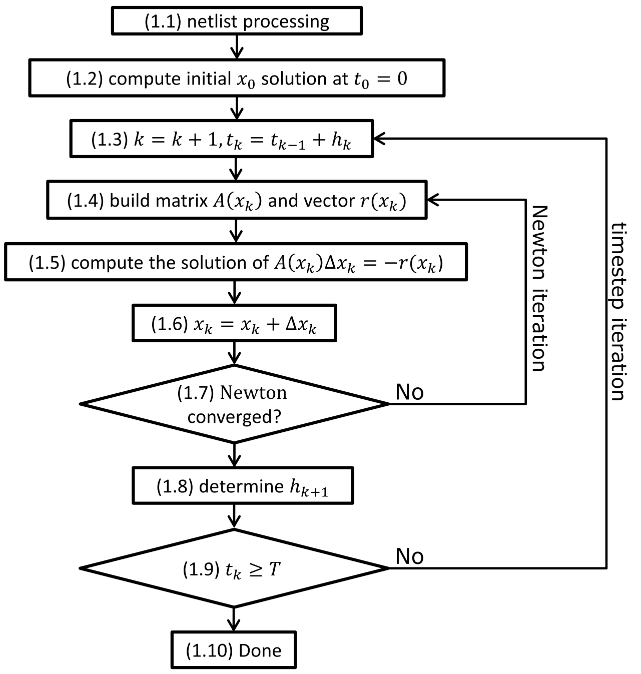

(1.1) netlist processing
(1.2) compute initial $x_0$ solution at $t_0 = 0$
(1.3) $k = k + 1, t_k = t_{k-1} + h_k$
(1.4) build matrix $A(x_k)$ and vector $r(x_k)$
(1.5) compute the solution of $A(x_k)\Delta x_k = -r(x_k)$
(1.6) $x_k = x_k + \Delta x_k$
(1.7) Newton converged?
No
Newton iteration
timestep iteration
(1.8) determine $h_{k+1}$
(1.9) $t_k \geq T$
No
(1.10) Done

**Figure 1 Transient analysis algorithm.** The flow diagram presents the algorithmic steps during the transient analysis. The steps (1.4)-(1.7) within the nested loop are the most critical in the transient analysis. The outer loop represents the time loop of the whole circuit.

partitions are overlapping and coupled explicitly by multiplicative or additive Schwarz iterations resulting in an accurate and fast simulation especially for RC dominated circuits. Another fast-SPICE approach [15] uses the hierarchical circuit description in the input netlist[a] to partition the circuit by capturing the repetitive elements in the circuits. If many instances of a given element share the same state, then significant computation can be saved during step (1.4) in Figure 1. Furthermore, depending on the coupling strength [15], selected partitions can be coupled explicitly. A similar approach is introduced in [16] with emphasis on efficient and parallel computing. The method presented in [17] also uses the hierarchical netlist structure for partitioning [6] and applies the multi-rate time integration for the resulted partitioning, resulting in a considerably shorter simulation time but also in unpredictable accuracy. For a comprehensive overview of fast-SPICE techniques we refer further to [10, 11].

Another important approach to further speed-up the simulation is to apply model-order reduction to the circuit used as input for the simulator such that accuracy is not compromised and simulation time is drastically reduced [10, 18, 19]. These methods are also named as SPICE-in, SPICE-out network reduction methods, and they work especially well with circuits dominated by passive elements [18, 19]. In an industrial environment they are typically applied before the actual SPICE-level simulations are started.

In order to deliver reliable SPICE accuracy, preserving the algorithmic framework in Figure 1 is an important aspect of the SPICE acceleration techniques. The main focus of these methods is to speed up and parallelize the inner loop of the algorithm, steps (1.4)-(1.7) in Figure 1. The direct linear solvers in SPICE simulators, depending on the nature of the circuit, scale $O(n^{1.3\text{-}1.8})$ with the size $n$ of the circuit [11]. Constructing the matrix and the right-hand side in step (1.4) is done with linear complexity $O(n^1)$, but the constant factor of the complexity function is of order $10^3$-$10^5$, since the industrial semiconductor models typically have numerous nonlinear equations to evaluate. All other steps ((1.6) and (1.7) in the inner loop) have linear $O(n^1)$ complexity and only require a couple of operations per MNA variable. Therefore for large circuits the solver step (1.5) is the most dominant part of the Newton loop and was the subject of several research work. In several publications [20–23] the authors present various methods to speed up and to parallelize the linear solver step in the inner loop. These approaches are limited not just by Amdahl's law for parallelization, but they also have limited speedup capability due to their pure linear algebra view on the problem [10]. Even the approaches that partition the circuit on the pure linear algebra view [24] of the problem [10, 11] have limited speed up capacity for different circuit types and sizes.

For mid-sized circuits,[b] the most compute-intensive part of the inner loop is the evaluation of the semiconductor models during the setup of the linear system (5), step (1.4) in Figure 1. Hence, other approaches target the evaluation of the semiconductor devices by either accelerated and parallelized evaluation [5, 25, 26] or by model order reduction such as table models [27] of the complex semiconductor models.

Focusing only on one single step of this inner loop results in significant speedup either for RC-[c] or semiconductor-dominated[d] circuits of either large or mid-sized circuits. For small[e] and mid-sized[f] circuits the semiconductor model evaluation and building the Jacobi matrix is usually the most dominant part. For large-sized circuits,[g] most of the computational workload is in the linear solver. In addition to the size $n$ of the circuit, it is also relevant which type of devices are dominating the circuit: If the circuit is dominated by parasitic resistors and capacitances and their number is usually considerably larger than $n$, then the Jacobi matrix becomes more dense and the solver more computationally dominant.

## 1.1 Structure of this paper

In this paper we present a comprehensive approach to speed up and parallelize all the computational intensive steps of the inner loop in Figure 1, while maintaining the single-rate characteristic and thus the reliable SPICE accuracy of the algorithm.

The starting point of our approach is a circuit partitioner. Our approach of partitioning uses, extends, and combines existing partitioning approaches. Similar to other domain decomposition approaches [6, 28] our partitioner minimizes the number of connection nodes between the partitions, but in addition it also makes sure that the fill-in rate of the resulting coupling system is limited [24] and that all partitions can be evaluated and solved in a fully parallel way [6]. This partitioner is presented in the first section of this paper.

In the second section of the paper, we present the resulting BBD matrix data structure and the hybrid solver that extends the approach presented in [24].

The third section of this paper describes the partition bypass approach that extends the BBD solver and the partition evaluation process. By skipping parts of step (1.4) and step (1.5) for converged partitions of the inner Newton loop in Figure 1 significant computation can be skipped, yet maintaining the same numerical precision of the simulation.

The final section of this paper presents the numerical results and simulation time comparison of our approach. We measure the run time and check the accuracy of our implementation for a large range of circuit types and circuit sizes. Thereby we demonstrate that our approach results in significant parallel and sequential speedup compared to classical SPICE algorithms and also delivers reliable accuracy for all circuits that we simulated.

## 2 Methods

### 2.1 Circuit partitioner

In most comprehensive SPICE and fast-SPICE acceleration approaches, the static or dynamic partitioner plays a central role. In our approach, since we keep the single-rate principle of the simulation and also for sake of simplicity, we only considered a static partitioner that divides the circuit in an early setup phase into a predefined number of partitions. In contrast to a static partitioner, dynamic partitioners re-partition the circuit during the transient analysis based on the current state and activity of each circuit part [15].

Due to the static nature of our partitioner it is crucial that all relevant performance aspects and constraints are considered already in the setup phase. We group the devices from the circuit into separate partitions as it is illustrated in Figure 2 for a simple circuit with only two partitions. The resulting partitions in Figure 2 have no electrical meaning because the partitioner minimizes the coupling nodes regardless of the subcircuit structure of the input circuit. Similar to common circuit partitioning approaches [6, 11, 17, 29], the devices are grouped such that the number of coupling nodes are minimized. This objective can be achieved with a general graph partitioning algorithm [30–32], where the devices are the nodes in the graph and the matrix entries in $A$ (see (5))[h] of the circuit represent the edges in the graph. Another approach is to transform the circuit into a hypergraph, and use an appropriate partitioning algorithm [29].

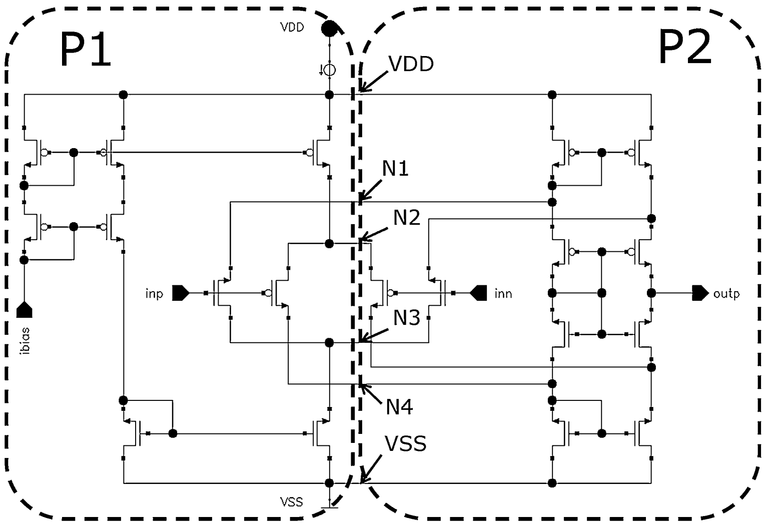


**Figure 2 Circuit partitioning.** The figure illustrates the partitioning of a simplified operational amplifier composed of 19 MOSFETs and one ideal voltage source of zero Volts into two partitions (P1 and P2). The six coupling nodes (VDD, N1, N2, N3, N4, and VSS) are marked by arrows. The first partition P1 has five internal nodes and one current variable (due to the MNA approach each voltage source has one current as MNA variable in the system), whereas the second partition P2 has six internal nodes. The whole circuit has 17 nodes (and thus 18 MNA unknowns) in total.

After grouping the devices into partitions and ordering the rows and columns of $A$ according to these partitions, the system matrix will have the BBD structure (6).

$$A = \begin{bmatrix} A_{1,1} & 0 & 0 & \cdots & 0 & A_{1,c} \\ 0 & A_{2,2} & 0 & \cdots & 0 & A_{2,c} \\ 0 & 0 & A_{3,3} & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & 0 & \cdots & 0 & A_{p,p} & A_{p,c} \\ A_{c,1} & A_{c,2} & A_{c,3} & \cdots & A_{c,p} & A_{c,c} \end{bmatrix}. \tag{6}$$

The block index $c$ represents the coupling part of the $p$ partitions in the circuit. One main objective of the partitioner is to create a matrix structure that can be built and partially solved independently by each partition. In this way, the coarser-grained parallelism can be realized significantly improving the parallel performance also for mid- and small-sized circuits. Since this BBD structure (6) is resulting from the device grouping partition approach, this also means that $A_{c,c}$ is the only matrix block that needs synchronization for writing in a multi-threaded mode. Fortunately, this write conflict can be simply solved by splitting the coupling system $A_{c,c}$ into the contributions from each partition, i.e.,

$$A_{c,c} = A_{c,c}^{(1)} + A_{c,c}^{(2)} + A_{c,c}^{(3)} + \cdots + A_{c,c}^{(p)}. \tag{7}$$

After splitting the coupling matrix $A_{c,c}$ the global system $A$ can be written also in this split form (7) with the following matrix structure:

$$\begin{bmatrix} A_{1,1} & 0 & 0 & \cdots & 0 & A_{1,c} \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 \\ A_{c,1} & 0 & 0 & \cdots & 0 & A_{c,c}^{(1)} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & A_{2,2} & 0 & \cdots & 0 & A_{2,c} \\ 0 & 0 & 0 & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & A_{c,2} & 0 & \cdots & 0 & A_{c,c}^{(2)} \end{bmatrix} + \cdots. \tag{8}$$

Analog to Figure 2 we denote the partition matrices by $A = A_{P1} + A_{P2} + \cdots + A_{Pp}$, where the global system matrix is the sum of all partition matrices.

Applying the same splitting to the right-hand-side vector, we get $p$ separate right-hand-side vectors $r(x_i) = r_{P1} + r_{P2} + \cdots + r_{Pp}$:

$$r(x_i) = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_p \\ r_c \end{bmatrix} = \begin{bmatrix} r_1 \\ 0 \\ \vdots \\ 0 \\ r_c^{(1)} \end{bmatrix} + \begin{bmatrix} 0 \\ r_2 \\ \vdots \\ 0 \\ r_c^{(2)} \end{bmatrix} + \cdots + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ r_p \\ r_c^{(p)} \end{bmatrix}. \tag{9}$$

Given the splittings (8) and (9), and the condition that each device must be assigned uniquely to one partition, there is no writing conflict for devices from different partitions, hence the partitions can build their matrices (step (1.4) in Figure 1) in a fully parallel and unsynchronized manner. In the upcoming section about partition bypass, we present further aspects of this matrix and right-hand-side separation, which will turn out to be beneficial also for sequential simulations.

The next crucial objective of our partitioning approach is to limit the fill-in entries in $A_{c,c}$ which will result from the LU solving process. A given matrix $A_{Pi}$ of partition $i$ is partially LU decomposed except the $A^{(i)}_{c,c}$ part. The solving step of the BBD matrix is presented in detail in the next section.

In previous approaches [6, 11] the authors point out that if these fill-ins are not controlled then for only $10^3$-$10^4$ coupling nodes solving the coupling system $A_{c,c}$ of the BBD matrix becomes a bottleneck or even unfeasible. In [24], the author uses a fill-in minimization technique by analyzing the elimination tree of the global BBD matrix and identifying the coupling nodes such that the fill-in entries are minimized. However, the resulting partitioning cannot be built independently by the devices, since this approach considers only the matrix view of the circuit.

Since it is difficult to compute the fill-in rate at the circuit device level, we analyze the fill-in rate of the coupling system in a second step after the device grouping, once the fill-in entries in $A^{(i)}_{c,c}$, $i = 1,\dots,p$ can be computed. In this second step, for each row of $A_{i,i}$ and $A_{i,c}$ we compute how many fill-in entries would be inserted into the matrix blocks $A_{c,i}$ and $A^{(i)}_{c,c}$. If the number of fill-ins for a row exceeds a threshold in the range of $[10^3, 10^4]$, then the row is moved from $A_{i,i}$ and $A_{i,c}$ to the coupling part $A_{c,i}$ and $A^{(i)}_{c,c}$. In our empirical tests it turned out that a constant threshold of 1500 increases the number of coupling nodes only marginally but decreases the number of fill-ins drastically in the accumulated $A_{c,c}$. The value of this threshold can be explained by the structure of the system matrix, where most of the matrix rows have less then 10 entries, therefore they cause marginal fill-in entries. According to [24] and to our experience, there are only proportionally few rows that cause a significant number of fill-ins, and these rows are detected by this threshold value and are moved to $A_{c,c}$.

The final objective of our partitioning approach is to ensure that all block matrices can be LU decomposed by a static pivoting solver. The fill-in minimization reordering does always symmetric row and column swapping, so that the diagonal elements remain on the diagonal. Node voltage MNA variables can be used for static pivoting, since they have nonzero diagonal entries. However, the branch current MNA variables which are required by voltage sources and inductors have no diagonal entries in all analyses. Therefore they require a neighboring node voltage MNA variable for pivoting. This row swapping between these two MNA variables must be possible within one partition or within the coupling part of the system. In this last step of the partitioning, it is ensured that all current MNA variables and their neighbor node voltage MNA variables are either in the same partition block matrix $A_{i,i}$ or in the coupling system $A_{c,c}$. For this reason, if necessary, further MNA variables are moved to the coupling system. In other words, we ensure that all current paths are contained completely either in a unique partition matrix or in the coupling matrix. In this way it is always ensured that the diagonal entries will stay nonzero during the block-wise Gaussian eliminations.

The novelty of our circuit partitioning approach is that we consider various aspects of the problem, that all enable at the end a fast and parallel simulation of mid-sized and large circuits. At the end of this section as a summary we list all the objectives that represent the core of our partitioning approach.

1. Parallel building and solving of each partition's matrix.
2. Minimize the number of coupling nodes.
3. Minimize the fill-in rate in the coupling system.
4. Ensure for all partitions the solvability with static pivoting LU solvers.

These objectives are achieved by the following three consecutive steps that form our novel partitioner's pseudo algorithm:

1. Group the devices into partitions with the graph partitioning algorithm [30] such that the coupling nodes are minimized.
2. Apply the fill-in threshold of 1500 to minimize the fill-in rate in the coupling system.
3. If necessary, ensure for all partitions the solvability with static pivoting LU solvers by moving MNA variables into the coupling part.

## 2.2 Solving the BBD system

In this section, we present in more detail our approach to solve the BBD system. For more details on the mathematical background of the BBD systems and Schur complement we refer to [11, 24, 33]. By permuting the rows and columns according to partition nodes $x_i$, $i = 1,\dots,p$ or coupling nodes $x_c$, the BBD system matrix $A$ and right-hand side $r$ take shape as

$$\begin{bmatrix} A_{1,1} & 0 & 0 & \cdots & 0 & A_{1,c} \\ 0 & A_{2,2} & 0 & \cdots & 0 & A_{2,c} \\ 0 & 0 & A_{3,3} & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & 0 & \cdots & 0 & A_{p,p} & A_{p,c} \\ A_{c,1} & A_{c,2} & A_{c,3} & \cdots & A_{c,p} & A_{c,c} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_p \\ x_c \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_p \\ r_c \end{bmatrix}. \tag{10}$$

The solution of each partition from (10), assuming $x_c$ is known, is given by $x_i = A_{i,i}^{-1}(r_i - A_{i,c}x_c)$, whereas the solution for the coupled system is determined by

$$\left(A_{c,c} - \sum_{i=1}^{p} A_{c,i}A_{i,i}^{-1}A_{i,c}\right)x_c = r_c - \sum_{i=1}^{p} A_{c,i}A_{i,i}^{-1}r_i. \tag{11}$$

Using the splitting of the partitioner for parallel matrix and right-hand side building, introduced in (8) and (9), the coupled system (11) is transformed to

$$\left(\sum_{i=1}^{p}\bigl(A_{c,c}^{(i)} - A_{c,i}A_{i,i}^{-1}A_{i,c}\bigr)\right)x_c = \sum_{i=1}^{p}\bigl(r_c^{(i)} - A_{c,i}A_{i,i}^{-1}r_i\bigr). \tag{12}$$

Each summand in the sums of (12) is the contribution from one partition, and only in the final step they need to be summed sequentially. Using the abbreviations

$$S_i = A_{c,c}^{(i)} - A_{c,i}A_{i,i}^{-1}A_{i,c}, \qquad s_i = r_c^{(i)} - A_{c,i}A_{i,i}^{-1}r_i,$$

the coupled system (12) can be written as

$$\left(\sum_{i=1}^{p} S_i\right)x_c = \sum_{i=1}^{p} s_i,$$

yielding the coupling system $Sx_c = s$ which needs to be solved before all $x_i$, $i = 1, \dots, p$ can be determined.

The contributions $S_i$ and $s_i$ of partitions $i = 1, \dots, p$ to the coupling matrix $S$ and right-hand side $s$ are computed by a partial in-place block LU-decomposition of each partition's matrix, i.e.,

$$\begin{bmatrix} A_{i,i} & A_{i,c} \\ A_{c,i} & A_{c,c}^{(i)} \end{bmatrix} = \begin{bmatrix} L_{i,i} & 0 \\ A_{c,i}U_{i,i}^{-1} & I \end{bmatrix} \begin{bmatrix} U_{i,i} & L_{i,i}^{-1}A_{i,c} \\ 0 & S_i \end{bmatrix}. \tag{13}$$

The matrix $S_i = A_{c,c}^{(i)} - A_{c,i}U_{i,i}^{-1}L_{i,i}^{-1}A_{i,c}$ will be the direct result of the partial block LU-decomposition, and the right-hand-side contribution is given by the subsequent partial forward substitution of the resulting LU-decomposition with the given right-hand side $s_i = r_c^{(i)} - A_{c,i}U_{i,i}^{-1}L_{i,i}^{-1}r_i$ for $i = 1, \dots, p$. Due to the criterion 4 of the partitioner it is ensured that the partial LU-decomposition is made with in-place and static pivoting solver, resulting in maximal numerical performance. Referring to Figure 1, we also point out that with our BBD approach not just step (1.4) but also partially step (1.5) can be computed in parallel. The first synchronization point in the Newton loop is the building and solving of the coupling system $Sx_c = s$.

Subsequent to the coupling system solution and in accordance to the partial LU decomposition (13), a partial backward substitution is necessary to compute the unknowns $x_i = U_{i,i}^{-1}L_{i,i}^{-1}(r_i - A_{i,c}x_c)$, $i = 1, \dots, p$. Once $x_c$ is known, this step can be computed in a parallel and unsynchronized manner.

Figure 3 summarizes the presented steps to build and to solve the BBD system. The three steps on Figure 3 represent steps (1.4) and (1.5) from Figure 1 which is the most computational intensive part of the Newton loop within transient simulations. The sequential part of Figure 3 is grouped into step (3.2) that needs to be addressed for both parallel and sequential computational performance.

## 2.3 Solving the coupling system

Building the matrix $S$ and corresponding right-hand side $s$ is a sequential but linearly complex task. Since the matrix $S$ is more dense than the partition matrices $A_{P_i}$, $i = 1, \dots, p$ [11, 24], solving this coupling system becomes more computational complex than building the

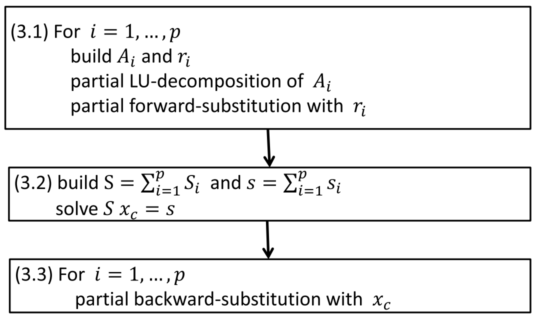

(3.1) For $i = 1, \dots, p$
build $A_i$ and $r_i$
partial LU-decomposition of $A_i$
partial forward-substitution with $r_i$
(3.2) build $S = \sum_{i=1}^{p} S_i$ and $s = \sum_{i=1}^{p} s_i$
solve $S\, x_c = s$
(3.3) For $i = 1, \dots, p$
partial backward-substitution with $x_c$

**Figure 3 Steps to build and solve the BBD system.** This figure illustrates in more detail steps (1.4) and (1.5) of the inner loop in Figure 1 in the frame of our BBD approach. Both of the loops, step (3.1) and step (3.3) of this figure can be done in fully parallel and decoupled mode, whereas step (3.2) represents the sequential bottleneck of the BBD approach.

matrix and right-hand side. Therefore, in the following subsection, we focus on our approach to solve the coupling system that is a crucial element in step (3.2) of Figure 3.

The criterion 3 of the partitioner ensures that the number of fill-ins in the matrix are reduced, but they are overall significantly higher than for the individual partition matrices.

For mid-sized circuits, if the number of coupling nodes is less than a few hundreds and a direct solver does not produce significant additional fill-ins, a direct solver works the best also for the coupling solver. However, even for mid-sized circuits, when the number of coupling nodes is beyond 300-400, efficient iterative solvers become more competitive than direct solvers.

Our approach is based on the ILU($\epsilon$)-preconditioned GMRES Krylov space algorithm, cf. [34–36]. This iterative solver was already successfully used in [24] as an efficient coupling system solver. While the author used a constant $\epsilon = 0.001$, we extend this approach by using an adaptive $\epsilon$-strategy. Previously in [24] this approach was tested for one constant matrix. During the transient simulation using the same threshold value $\epsilon$, as activity might change in the circuit, the pattern of the ILU($\epsilon$) preconditioner might also change significantly. On the other hand, recomputing the pattern of the ILU($\epsilon$) preconditioner poses a significant computational overhead. Therefore the convergence of the preconditioned GMRES and the computational overhead for an ILU($\epsilon$) update needs to be balanced. If the pattern stays the same, then one simple measure is to only update the ILU-decomposition of the matrix, considering the current values of the Newton iteration matrix.

In summary, we developed an adaptive $\epsilon$-strategy for our ILU preconditioner which detects poor convergence or divergence of the GMRES by monitoring the number of required iterations. Depending on the level of divergence one of the following steps is being executed:

1. Update the ILU-decomposition using the current pattern and values of $A$.
2. Recompute the pattern of ILU($\epsilon$) with the value of $\epsilon$ and do (1).
3. Decrease $\epsilon$ by two and do (2).

We use a starting value of $\epsilon = 0.05$ which for average simulations remains unchanged or decreases only slightly, so that the ILU($\epsilon$) preconditioner remains computationally cheap. The values of $\epsilon$ are limited by a lower bound of $10^{-5}$, and in case of divergence of the iterative solver for $\epsilon \leq 10^{-5}$ a direct solver is used instead. With this backup strategy we do not just ensure robustness of transient simulations but we also avoid too small values of $\epsilon$.

One additional important aspect is that the core GMRES method consists mainly of matrix vector multiplications [35] which can be parallelized efficiently, but the ILU preconditioner, for such matrix sizes, does not run efficiently in parallel. Therefore, large values of $\epsilon$ favor parallel simulation since the computations in ILU($\epsilon$) are marginal compared to GMRES. On the other hand, with small values of $\epsilon$ a more accurate preconditioner is created and fewer GMRES iterations might be required. For the enlisted reasons it is crucial to automatically select the optimal values of $\epsilon$ for a given transient simulation and circuit, extracting the maximal efficiency for sequential and parallel transient simulations.

## 2.4 Partition bypass acceleration

In the previous two sections, we presented the main components of our approach. The partitioner, the parallel matrix building and solving method result already in substantial transient simulation speedups. In this section, we present one additional acceleration method that exploits the latency in the partitioned circuit while maintaining the single-rate and SPICE-accuracy aspects of the transient simulation. Furthermore, this acceleration technique is built on top of the previous methods and can be deactivated.

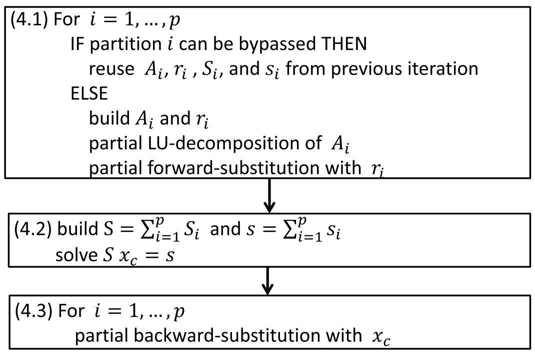
The characteristic of large- and mid-sized circuits is that during their operation mode the activity is mainly concentrated within a few number of partitions [10, 11]. In this analog context, 'activity' means the process when a semiconductor changes its state from ON to OFF or vice versa. During this transition the semiconductors have highly nonlinear behavior and SPICE-level accuracy is crucial to capture this transition. Furthermore, such nonlinear transitions can trigger a chain of other transitions at the same time during the Newton loop. Therefore it is crucial for SPICE-level accuracy to simulate the partitions in a single-rate way.

The main idea of the partition bypass is to reuse the factorized $A_{i,i}$, the right-hand side $r_i$, and the contributions to the coupling system $S_i$ and $s_i$ from the previous Newton iteration, if the bypass criterions are met. Since we operate directly with the partition matrix $A_{i,i}$ and with the right-hand side $r_i$, as the Newton linearization (5) shows, we cannot reuse the matrices from previous time steps, while the integration coefficient $\alpha$ changes with the time step size and integration method. Therefore the bypass method starts with the second Newton iteration, such that $A_i$, $r_i$, $S_i$ and $s_i$ are computed with the correct $\alpha$. Hence, each device is evaluated at least once in a time step ensuring the single-rate aspect of our approach.

Figure 4 presents the modified Newton loop of Figure 3. The difference to Figure 3 is only in step (4.1). If for a partition $i$ the bypass criterion is fulfilled, then the whole computation is skipped and the results from the previous Newton loop are reused. Steps (4.2) and (4.3) are computed as in Figure 3. If a partition is bypassed for a Newton iteration, this does not imply that for subsequent Newton iterations it will be skipped as well, since in step (4.2) $x_c$ is updated and in step (4.3) even a bypassed partition's unknowns $x_i$ will be changed.

As next we introduce the partition bypass criterion that is the key for the success of this acceleration. A strict partition bypass criterion increases the speedup only marginally, whereas inaccurate criterions can cause convergence problems or can even produce wrong results in certain cases.

We denote by $\hat{x}_i$ the values of the unknowns $x_i$ of partition $i$ where the last $A_{i,i}$ and $r_i$ was built. Since not all of the elements $x_c$ contribute to all partitions, we denote by $x_c^{(i)}$ the



```
(4.1) For $i = 1, \dots, p$
        IF partition $i$ can be bypassed THEN
          reuse $A_i$, $r_i$, $S_i$, and $s_i$ from previous iteration
        ELSE
          build $A_i$ and $r_i$
          partial LU-decomposition of $A_i$
          partial forward-substitution with $r_i$

(4.2) build $S = \sum_{i=1}^{p} S_i$ and $s = \sum_{i=1}^{p} s_i$
      solve $S\,x_c = s$

(4.3) For $i = 1, \dots, p$
        partial backward-substitution with $x_c$
```

**Figure 4 Partition bypass acceleration.** If the bypass criterion for a partition is met, the devices within this partition need not to be evaluated and matrix LU-decomposition can also be skipped. Subsequent computations are unchanged compared to Figure 3.

subset of $x_c$ that contributes to partition $i$, and $\hat{x}_c^{(i)}$ denotes the last values of $x_c^{(i)}$ where the partition $i$ was evaluated. Newton convergence of partition $i$ is monitored by the vector $T_i$ which has the values

$$T_{i;j} = \frac{|r_{i;j}|}{ABSTOL + RELTOL \cdot \mathrm{imax}_{i;j}}, \quad j \in Pi. \tag{14}$$

Here $\mathrm{imax}_{i;j}$ is the absolute value of the maximal current contribution to voltage node $j$ within partition $i$, and $Pi$ represents the MNA unknowns $x_i$ and those MNA unknowns from $x_c$ that contribute to partition $i$. The absolute tolerance $ABSTOL$ is set to $10^{-12}$ for the voltage nodes and to $10^{-6}$ for the current equations. The relative tolerance $RELTOL$ is $10^{-3}$ for all MNA unknowns. The indicator vector represents the convergence state of the MNA variables during the Newton loop. The right-hand-side vector is the sum of the currents at a given node, and this sum, according to Kirchhoff's current law, should converge numerically to zero. If $T_{i;j} \leq 1.0$, then the $j$th unknown of partition $i$ is converged in terms of the Newton criterion. The Newton iteration is converged for partition $i$ when $\|T_i\|_\infty \leq 1.0$. With the introduced notation we define the bypass indicator for all MNA variables belonging to a partition $i = 1, \dots, p$:

$$B_i = \begin{cases} 1, & \text{if } (\|\hat{x}_i - x_i\|_\infty < \epsilon_x \text{ and } \|\hat{x}_c^{(i)} - x_c^{(i)}\|_\infty < \epsilon_x) \text{ or } \|T_i\|_\infty < \epsilon_N, \\ 0, & \text{otherwise,} \end{cases} \tag{15}$$

where $\epsilon_x = 10^{-6}$ and $\epsilon_N = 0.8$. The tolerance $\epsilon_x$ represents the trust region of the last evaluation point $\hat{x}_i$ and $\hat{x}_c^{(i)}$ where partition $i$ was evaluated. The coefficient $\epsilon_N \leq 1.0$ ensures that the partition is converged in terms of the Newton criterions. The indicator $B_i$ in (15) has a value of 1 when the new evaluation point for partition $i$ is within the trust region of the last evaluation point or when the Newton is already converged for partition $i$. If the value of the indicator function $B_i$ is zero, it means that the partition $i$ should not be bypassed.

The bypass indicators (15) are evaluated at the beginning of each Newton iteration. The computational overhead for this block bypassing consists in storing the partition's LU-factorized matrix and the right-hand side, and in computing the bypass indicators (15). Thus it does not pose significant additional computations compared to Figure 3.

The bypass method of Figure 4 transforms the Newton method into a Quasi-Newton method by using a reduced model for the partition. If a partition is bypassed, then a constant extrapolation is used with the values of the last evaluation point. The possibility of linear extrapolation has also been studied in [37], but it turned out that the constant extrapolation gives the best cost-benefit ratio overall for robust transient simulations.

### 2.5 Applicability of the BBD solver for other analyses

So far we investigated the concept and application of our BBD matrix and solver approach in the context of transient analyses. In the following we emphasize some of the high-level aspects of our approach applied to other circuit analyses that are often used beside transient analysis. The basic principle that each device contributes additively to the overall system is still true for DC and AC analyses. Therefore our concept of partitioning the circuit and solving the respective BBD system can be correspondingly used also for these analyses.

Apart from the absence of dynamic contributions, each step of a DC analysis can be considered as computing one transient timestep. Therefore the solving approach of the whole BBD system is the same as for transient analyses, and it might result in substantial performance gains compared to conventional solvers.

A small-signal AC or AC NOISE analysis is built upon a linearization around an operating point and results in a sequence of linear problems for the selected frequency points. Then the whole BBD system becomes complex-valued, but is still structurally equivalent to the respective transient system.

In both cases we solve the coupling system of the BBD matrix with a direct solver, allowing to use the same matrix structure and solver for all these analyses. Further detailed elaboration of our matrix and solver approach in the context of AC and DC analysis is beyond the scope of this paper.

## 3 Results and discussion

In this section, we demonstrate the speedup potential of our presented method to significantly speed up transient analog circuit simulations. The implementation of the presented approach was done in Infineon's in-house SPICE simulator TITAN, and all the numerical comparisons were made in the frame of this simulator, assuring that the numerical methods are tested in the same environment. The TITAN simulator is used in productive environment and its implementation is trimmed for high performance computing, therefore the presented novel method is tested in practical productive environment, showing the true potential of our approach.

For the numerical comparison we consider the nine circuits from Table 1. These circuits represent a wide range of circuit sizes and types. We consider the semiconductor-dominated circuits cir1, cir2, cir3, cir6, cir7, and cir8, whereas the RC-dominated circuits are cir4, cir5, and cir9. Furthermore we consider the mid-sized circuits where the total number of MNA variables is less than $10^6$, these circuits are cir1, ..., cir5, whereas the large circuits are cir6, ..., cir9. As shown in Table 1, the number of nonzero entries in the system matrix is increasing with the increasing number of MNA unknowns.

In an industrial context the RC-dominated circuits are always preprocessed by a state-of-the-art SPICE-in, SPICE-out network reduction tool [18, 19]. This is also the case for the input circuits in our test suite. Hence the speedup of such network reduction techniques

**Table 1 List of the nine test circuits that were taken from a wide range of applications (e.g., ADCs, mixers, PLLs) and semiconductor technologies**

| Name | # semicond. | # MOSFET | # R | # C | # MNA | # NNZ |
|---|---|---|---|---|---|---|
| cir1 | 35,671 | 35,621 | 1160 | 1633 | 20,826 | 173,347 |
| cir2 | 20,057 | 20,057 | 395 | 2820 | 9837 | 92,921 |
| cir3 | 10,979 | 10,919 | 599 | 34,010 | 6559 | 90,859 |
| cir4 | 713 | 613 | 11,585 | 26,533 | 10,492 | 93,762 |
| cir5 | 31,185 | 31,075 | 4884 | 205,219 | 80,706 | 885,501 |
| cir6 | 239,034 | 217,034 | 4806 | 13,348 | 145,903 | 1,158,732 |
| cir7 | 319,395 | 318,395 | 4072 | 43,837 | 170,524 | 1,498,263 |
| cir8 | 109,379 | 109,279 | 5563 | 745,088 | 270,044 | 3,129,289 |
| cir9 | 55,601 | 55,491 | 319,110 | 2,228,295 | 432,009 | 5,991,033 |

For each circuit we list in the second column the total number of semiconductors (MOSFETs, BJTs, JFETs, diodes) and in the next column only the number of MOSFETs. In the following columns we enlist the number of resistors and capacitors which are the dominant part for extracted circuits. In the last two columns we list the total number of MNA variables in the DAE system and the number of nonzero entries in the resulting system matrix.

is orthogonal to the speedup of actual SPICE-like simulations, and not taken into account in the following comparison.

As a base line of comparison we choose the KLU solver [23, 38] within the TITAN simulator. This solver is well suited for sparse matrices that arise from analog circuit simulation, and this solver is also widely used in open-source [4, 5] and commercial SPICE simulators. Since KLU [23, 38] does not have a parallel version, we consider the elapsed time of the sequential simulation as a reference for the sequential and parallel simulations with our approach.

The measured elapsed time results of the sequential and parallel simulations are presented in Table 2. The number of partitions and the bypass rate of the partitions are presented in Table 3. We also underline here that the presented circuits and their setup are taken from an industrial and practical context. Therefore they contain output and other sequential parts that by Amdahl's law limits the theoretical speedup of the total simulation time. For these reasons, we only scale up to 8 CPUs and we aim at a speedup in a range of $[3,4]$ with 8 CPUs compared to the same sequential run.

In the first step we consider the mid-sized circuits, cir1, ..., cir5. Within this group there are both RC- and semiconductor-dominated circuits. cir1, cir2, and cir3 are semiconductor-dominated and our approach matches the sequential performance of the KLU solver. In these cases the partition bypass strategy also significantly improves the BBD solver performance. Especially for cir1 and cir3 we get a partition bypass rate around 30% (see Table 3) and thus additional overall sequential speedup of 20% in the elapsed time. The parallel scaling for these circuits is also satisfactory since the speedup is around factor 3 with 8 CPUs.

However for RC dominated circuits, cir4-5, the parallel scaling only reaches a factor of 2 with 8 CPUs. For such circuits, as Table 3 shows, the coupling system is relatively large compared to the overall system matrix, and the presented ILU preconditioned GMRES solver for the coupling system is not well suited for parallelization. Furthermore, the evaluation of simple devices such as linear resistors and capacitors represents a significantly smaller and parallelizable computation task than complex semiconductor models. For these reasons, the speedup factor compared to the semiconductor-dominated circuits is significantly lower. In comparison to the KLU solver, our BBD solver performs significantly better for cir5, but for cir4, due to the small circuit size, KLU performs sequentially 20% better.

The large-sized circuits truly display the true potential of our approach. For the semiconductor-dominated circuits cir6 and cir7 the speedup compared to KLU is substantial, and additionally we get more than factor 4 speedup with 8 CPUs. Overall, in comparison with the sequential KLU solver in both cases we get double digit factor speedup factors, which yield a huge step forward in the analog transient circuit simulation.

For the RC-dominated large circuits the sequential speedup of the BBD compared to the KLU is substantial and for very large circuits can have double digits. On the other hand in these cases the parallel speedup of the BBD solver is only around factor 2. This bad behavior is due to the ILU preconditioner which runs sequentially at the moment. For such very large circuits this limits the parallel scalability of our approach.

Another important aspect is the effect of the partition bypassing. As it is shown in Table 2, turning on the partition bypassing in sequential mode does not have any significant effect on the number of time steps nor on the number of iterations. Therefore

**Table 2 List of the 9 test circuits that were taken from concrete circuits**

| Circuit/solver/# CPU | Elapsed time (s) | # time steps | # Newton iter. | Speedup |
|---|---|---|---|---|
| cir1/KLU /1 | 55.05 | 400 | 1781 | |
| cir1/BBD1/1 | 52.33 | 422 | 1861 | 1.05 |
| cir1/BBD2/1 | 45.45 | 418 | 1895 | 1.21 |
| cir1/BBD2/2 | 29.44 | 423 | 1929 | 1.87 |
| cir1/BBD2/4 | 18.59 | 419 | 2026 | 2.86 |
| cir1/BBD2/8 | 16.43 | 412 | 2013 | 3.35 |
| cir2/KLU /1 | 42.72 | 657 | 2450 | |
| cir2/BBD1/1 | 43.82 | 657 | 2438 | 0.97 |
| cir2/BBD2/1 | 42.43 | 686 | 2545 | 1.01 |
| cir2/BBD2/2 | 27.53 | 655 | 2453 | 1.55 |
| cir2/BBD2/4 | 19.16 | 707 | 2574 | 2.22 |
| cir2/BBD2/8 | 14.30 | 659 | 2467 | 2.98 |
| cir3/KLU /1 | 165.95 | 1920 | 7803 | |
| cir3/BBD1/1 | 139.00 | 1844 | 7452 | 1.19 |
| cir3/BBD2/1 | 118.44 | 1870 | 7673 | 1.40 |
| cir3/BBD2/2 | 83.88 | 1872 | 7646 | 1.97 |
| cir3/BBD2/4 | 62.10 | 1877 | 7756 | 2.67 |
| cir3/BBD2/8 | 55.92 | 1845 | 7655 | 2.96 |
| cir4/KLU /1 | 125.01 | 3206 | 10,257 | |
| cir4/BBD1/1 | 153.48 | 3213 | 10,488 | 0.81 |
| cir4/BBD2/1 | 151.89 | 3218 | 10,488 | 0.82 |
| cir4/BBD2/2 | 103.58 | 3213 | 10,488 | 1.21 |
| cir4/BBD2/4 | 87.31 | 3213 | 10,488 | 1.43 |
| cir4/BBD2/8 | 71.39 | 3213 | 10,488 | 1.75 |
| cir5/KLU /1 | 1616.62 | 892 | 2987 | |
| cir5/BBD1/1 | 510.28 | 897 | 3032 | 3.16 |
| cir5/BBD2/1 | 508.44 | 897 | 3032 | 3.18 |
| cir5/BBD2/2 | 354.05 | 897 | 3030 | 4.56 |
| cir5/BBD2/4 | 268.39 | 897 | 3031 | 6.02 |
| cir5/BBD2/8 | 256.78 | 897 | 3034 | 6.30 |
| cir6/KLU /1 | 2482.57 | 153 | 893 | |
| cir6/BBD1/1 | 225.35 | 154 | 880 | 11.01 |
| cir6/BBD2/1 | 212.00 | 147 | 866 | 11.71 |
| cir6/BBD2/2 | 123.13 | 147 | 869 | 20.16 |
| cir6/BBD2/4 | 81.50 | 147 | 875 | 30.46 |
| cir6/BBD2/8 | 66.96 | 147 | 880 | 37.10 |
| cir7/KLU /1 | 2456.89 | 176 | 1132 | |
| cir7/BBD1/1 | 497.99 | 176 | 1132 | 4.93 |
| cir7/BBD2/1 | 475.30 | 180 | 1164 | 5.16 |
| cir7/BBD2/2 | 286.65 | 180 | 1164 | 8.57 |
| cir7/BBD2/4 | 172.19 | 180 | 1164 | 14.29 |
| cir7/BBD2/8 | 132.23 | 180 | 1164 | 18.58 |
| cir8/KLU /1 | 8490.84 | 484 | 2964 | |
| cir8/BBD1/1 | 2593.06 | 454 | 2829 | 3.27 |
| cir8/BBD2/1 | 2462.75 | 463 | 2879 | 3.44 |
| cir8/BBD2/2 | 1846.18 | 462 | 2877 | 4.60 |
| cir8/BBD2/4 | 1470.81 | 464 | 2872 | 5.77 |
| cir8/BBD2/8 | 1312.55 | 458 | 2854 | 6.47 |
| cir9/KLU /1 | 23,238.25 | 68 | 193 | |
| cir9/BBD1/1 | 1296.32 | 68 | 193 | 17.92 |
| cir9/BBD2/1 | 1254.30 | 68 | 209 | 18.52 |
| cir9/BBD2/2 | 1083.28 | 68 | 209 | 21.45 |
| cir9/BBD2/4 | 870.81 | 68 | 209 | 26.68 |
| cir9/BBD2/8 | 712.55 | 68 | 209 | 32.61 |

The base line for comparison is a simulation with the KLU solver [23, 38]. The sequential run with the BBD1 solver is the presented approach but without partition bypassing. The sequential and parallel runs with BBD2 solver represent the full approach with partition bypassing. For each sequential and multi-threaded simulation we enlist the elapsed simulation time on an Intel Xeon 2.9 GHz processor with 12 cores, without considering any setup time of the solver or simulator. Starting from the third column, we enlist the number of time steps, the number of Newton iterations, and the speedup in the elapsed time compared to the KLU solver.

**Table 3 The resulting number of partition and bypass ratio for each of the nine test circuits**

| Circuit | # partitions | Partition bypass ratio | # coupling nodes | # NNZ in *S* |
|---|---|---|---|---|
| cir1 | 16 | 33% | 1005 | 4.8E+4 |
| cir2 | 16 | 12% | 797 | 5.3E+4 |
| cir3 | 16 | 29% | 831 | 6.6E+4 |
| cir4 | 16 | 9% | 1003 | 5.7E+4 |
| cir5 | 24 | 10% | 6402 | 5.5E+5 |
| cir6 | 48 | 14% | 6550 | 3.7E+5 |
| cir7 | 56 | 12% | 7277 | 4.9E+5 |
| cir8 | 64 | 4% | 20,137 | 1.9E+6 |
| cir9 | 64 | 11% | 90,347 | 1.3E+7 |

In the last two columns we enlist the number of coupling nodes and the number of nonzero entries in the coupling matrix $S$.

the presented approach for partition bypass represents a robust method to speed up the presented BBD solver. Table 3 summarizes the partition bypass rate which depends not just on the size of the circuit or number of partitions, but mostly on the scenario that is simulated. A circuit start-up scenario usually results in single digit partition bypass rates, since the supply voltage ramp up usually affects the whole circuit and results in activity in all partitions. On the other hand for circuits in the normal working regime at a given time point the activity is concentrated in a relatively small portion of the circuit.

In this section we demonstrated that the presented approach for transient analog circuit simulation is capable of double digit speedups for large-sized circuits in comparison to the state-of-the-art KLU solver.

## 4 Conclusion and outlook

In this publication we presented a holistic approach to achieve double digit speedups for analog transient simulation of large-sized circuits compared to existing state-of-the-art KLU [23, 38] solver. The novelty of our approach consists in the combination and extension of existing approaches in a unique and unprecedented way. In the first step of our approach we partition the circuit such that the system matrix and right-hand side can be built in parallel, while minimizing the fill-in rate in the resulting coupling system of the BBD matrix. For solving the coupling system during the simulation we introduced an adaptive $\epsilon$ approach for the ILU($\epsilon$) preconditioned GMRES solver which solves the coupling part efficiently. As an additional speedup measure we introduced the partition bypass method. If certain criterions are met during the Newton iteration of a time step, the partition bypass method skips significant computations of the Newton loop. The numerical examples clearly underline not just the robustness of our approach but its true speedup potential especially for large-sized circuits in the frame of an industrial analog simulator.

Further work should be focused on the ILU($\epsilon$) preconditioner of the GMRES solver which represents a performance bottleneck for large-sized circuits. Furthermore the GMRES solver should be coupled to the Newton convergence criterion such that fewer GMRES iterations are computed. The current form of the partition bypassing is also rather simple and could be also further extended to increase the bypass ratio of the partitions.

**Acknowledgements**
We acknowledge the support of Infineon Technologies AG for this publication.

**Funding**
This research was completely supported by Infineon Technologies AG.

**Competing interests**
The authors declare that they have no competing interests.

**Authors' contributions**
The main idea of this paper was proposed by JB and the implementation was done also by JB. GD and KW significantly supported JB in the implementation and helped preparing the manuscript. All authors read and approved the final manuscript.

**Endnotes**

[a] Netlist is the input file format that describes the hierarchical structure of the circuit.
[b] $10^4 \leq n \leq 10^6$.
[c] Circuit where the linear resistors and capacitance dominate the circuit. This type of circuit is also called post-layout.
[d] The number of MOSFETs, bipolar transistors, diodes, and other semiconductors are dominant in the circuit. This is also called pre-layout circuit.
[e] $n \leq 10^3$.
[f] $10^3 < n \leq 10^6$.
[g] $n > 10^6$.
[h] For the sake of simplicity we use $A = A(x_i)$.

## Publisher's Note

Received: 21 July 2017 Accepted: 30 November 2017 Published online: 08 December 2017

**References**

1. Nagel LW, Pederson DO. SPICE (simulation program with integrated circuit emphasis). Berkeley: EECS Department, University of California; 1973. Technical Report UCB/ERL M382. http://www2.eecs.berkeley.edu/Pubs/TechRpts/1973/22871.html.
2. Cadence Design Systems, Inc. Spectre user manual. Cadence Design Systems, Inc. 2017. https://www.cadence.com/content/cadence-www/global/en_US/home/tools/custom-ic-analog-rf-design/circuit-simulation/spectre-circuit-simulator.html.
3. Synopsys, Inc. HSPICE user manual. Synopsys, Inc. 2017. https://www.synopsys.com/verification/ams-verification/circuit-simulation/hspice.html.
4. Nenzi P, Vogt H. Ngspice user manual. 2014. http://ngspice.sourceforge.net/docs/ngspice26-manual.pdf.
5. Keiter ER, Aadithya KV, Mei T, Russo TV, Schiek RL, Sholander PE, Thornquist HK, Verley JC. Xyce parallel electronic simulator reference guide. 2016. https://xyce.sandia.gov/downloads/_assets/documents/Reference_Guide.pdf.
6. Fröhlich N, Riess BM, Wever UA, Zheng Q. A new approach for parallel simulation of VLSI circuits on a transistor level. IEEE Trans Circuits Syst I, Fundam Theory Appl. 1998;45(6):601-13.
7. Feldmann U, Schultz R. TITAN: a universal circuit simulator with event control for latency exploitation. In: Fourteenth European solid-state circuits conference, ESSCIRC 88. 1988. p. 183-5.
8. Feldmann U, Wever UA, Zheng Q, Schultz R, Wriedt H. Algorithms for modern circuit simulation. Arch Elektron Übertragtech. 1992;46(4):274-85.
9. Günther M, Feldmann U, ter Maten J. Modelling and discretization of circuit problems. In: Schilders WHA, ter Maten EJW, editors. Handbook of numerical analysis, vol. XIII. Special volume: numerical methods in electromagnetics. Amsterdam: Elsevier; 2005. p. 523-659.
10. Rewienski M. A perspective on fast-SPICE simulation technology. Simulation and verification of electronic and biological systems. Berlin: Springer; 2011. doi:10.1007/978-94-007-0149-6-2.
11. Li P. Parallel circuit simulation: a historical perspective and recent developments. Found Trends Electron Des Autom. 2012;5(4):211-318. doi:10.1561/1000000020.
12. Deng A-C. On network partitioning algorithm of large-scale CMOS circuits. IEEE Trans Circuits Syst. 1989;36:294-9. doi:10.1109/31.20209.
13. Deng A-C, Tuan JF, Ong LW. An investigation on parasitic couplings and feedback loops in the CMOS circuits. In: 1989 IEEE international symposium on circuits and systems. Vol. 2. 1989. p. 864-7. doi:10.1109/ISCAS.1989.100488.
14. Peng H, Cheng C-K. Parallel transistor level circuit simulation using domain decomposition methods. In: Proceedings of the 2009 Asia and South Pacific design automation conference, ASP-DAC '09. Piscataway: IEEE Press; 2009. p. 397-402. http://dl.acm.org/citation.cfm?id=1509633.1509732.
15. Tcherniaev A, Feinberg I, Chan W, Tuan JF, Deng AC. Transistor level circuit simulator using hierarchical data. Google Patents. 2003. US Patent 6,577,992. https://www.google.es/patents/US6577992.
16. Lin PS, Wen KS, Perng RK. Simulation of circuits with repetitive elements. Google Patents. 2014. US Patent 8,832,635. https://www.google.com/patents/US8832635.
17. Striebel M. Hierarchical mixed multirating for distributed integration of DAE network equations in chip design [PhD thesis]. University of Wuppertal, Department of Applied Mathematics and Numerical Analysis; 2006.
18. Kerns KJ, Yang AT. Stable and efficient reduction of large, multiport RC networks by pole analysis via congruence transformations. IEEE Trans Comput-Aided Des Integr Circuits Syst. 1997;16:734-44.
19. Ionutiu R, Rommes J, Schilders WHA. SparseRC: sparsity preserving model reduction for RC circuits with many terminals. IEEE Trans Comput-Aided Des Integr Circuits Syst. 2011;30:1828-41.

20. Chen X, Wang Y, Yang H. NICSLU: an adaptive sparse matrix solver for parallel circuit simulation. IEEE Trans Comput-Aided Des Integr Circuits Syst. 2013;32(2):261-74.
21. Chen X, Wu W, Wang Y, Yu H, Yang H. An EScheduler-based data dependence analysis and task scheduling for parallel circuit simulation. IEEE Trans Circuits Syst II, Express Briefs. 2011;58(10):702-6. doi:10.1109/TCSII.2011.2164148.
22. Chen X, Ren L, Wang Y, Yang H. GPU-accelerated sparse LU factorization for circuit simulation with performance modeling. IEEE Trans Parallel Distrib Syst. 2015;26(3):786-95.
23. Davis TA, Palamadai Natarajan E. Algorithm 907: KLU, a direct sparse solver for circuit simulation problems. ACM Trans Math Softw. 2010;37(3):Article No. 36. doi:10.1145/1824801.1824814.
24. Bomhof CW. Iterative and parallel methods for linear systems with application in circuit simulation [PhD thesis]. University Utrecht, Computer Science Department; 2001.
25. Kapre N, DeHon A. Accelerating SPICE model-evaluation using FPGAs. In: 2009 17th IEEE symposium on field programmable custom computing machines, FCCM '09. 2009. doi:10.1109/FCCM.2009.14.
26. Bayoumi AM, Hanafy YY. Massive parallelization of SPICE device model evaluation on GPU-based SIMD architectures. In: Proceedings of the 1st international forum on next-generation multicore/manycore technologies, IFMT '08. New York: ACM; 2008. Article No. 12. doi:10.1145/1463768.1463784.
27. Vlach M. Modeling and simulation with Saber. In: Proceedings of the third annual IEEE ASIC seminar and exhibit. 1990. doi:10.1109/ASIC.1990.186080.
28. Yu H, Chu C, Shi Y, Smart D, He L, Tan SX-D. Fast analysis of a large-scale inductive interconnect by block-structure-preserved macromodeling. IEEE Trans Very Large Scale Integr (VLSI) Syst. 2010;18(10):1399-411.
29. Miettinen P, Honkala M, Roos J. Using METIS and hMETIS algorithms in circuit partitioning. 2006.
30. Karypis G, Kumar V. METIS - unstructured graph partitioning and sparse matrix ordering system, version 2.0. Minneapolis: Department of Computer Science, University of Minnesota; 1995. Technical report.
31. Chevalier C, Pellegrini F. PT-Scotch: a tool for efficient parallel graph ordering. Parallel Comput. 2008;34(6-8):318-31. doi:10.1016/j.parco.2007.12.001.
32. Devine KD, Boman EG, Riesen LA, Catalyurek UV, Chevalier C. Getting started with Zoltan: a short tutorial. In: Proceedings of the 2009 Dagstuhl seminar on combinatorial scientific computing. 2009. Also available as Sandia National Labs Technical Report SAND2009-0578C.
33. Basermann A, Cortial-Goutaudier F, Jaekel U, Hachiya K. Parallel solution techniques for sparse linear systems in circuit simulation. Math Ind. 2004;4:112-9. doi:10.1007/978-3-642-55872-6_10.
34. Saad Y. Iterative methods for sparse linear systems. 2nd ed. Philadelphia: Society for Industrial and Applied Mathematics; 2003.
35. Saad Y, Schultz MH. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J Sci Stat Comput. 1986;7(3):856-69. doi:10.1137/0907058.
36. Davis TA. Direct methods for sparse linear systems. Philadelphia: Society for Industrial and Applied Mathematics; 2006. doi:10.1137/1.9780898718881.bm.
37. Syed AA. Fast quasi-Newton method for partitioned analog circuit simulation in time domain [master's thesis]. Technische Universität München, Computer Science Department; 2016.
38. Davis TA, Natarajan EP. Sparse matrix methods for circuit simulation problems. Math Ind. 2012;16:3-14. doi:10.1007/978-3-642-22453-9_1.