Parallel Preconditioned Hierarchical Harmonic Balance for Analog and RF Circuit Simulation

Peng Li[1] and Wei Dong[2]
[1]*Department of Electrical and Computer Engineering, Texas A&M University*
[2]*Texas Instruments*
USA

1. Introduction

Circuit simulation is a fundamental enabler for the design of integrated circuits. As the design complexity increases, there has been a long lasting interest in speeding up transient circuit simulation using paralellization (Dong et al., 2008; Dong & Li, 2009b;c; Reichelt et al., 1993; Wever et al., 1996; Ye et al., 2008).
On the other hand, Harmonic Balance (HB), as a general frequency-domain simulation method, has been developed to directly compute the steady-state solutions of nonlinear circuits with a periodic or quasi-periodic response (Kundert et al., 1990). While being algorithmically efficient, densely coupling nonlinear equations in the HB problem formulation still leads to computational challenges. As such, developing parallel harmonic balance approaches is very meaningful.
Various parallel harmonic balance techniques have been proposed in the past, e.g. (Rhodes & Perlman, 1997; Rhodes & Gerasoulis, 1999; Rhodes & Honkala, 1999; Rhodes & Gerasoulis, 2000). In (Rhodes & Perlman, 1997), a circuit is partitioned into linear and nonlinear portions and the solution of the linear portion is parallelized; this approach is beneficial if the linear portion of the circuit analysis dominates the overall runtime. This approach has been extended in (Rhodes & Gerasoulis, 1999; 2000) by exposing potential parallelism in the form of a directed acyclic graph. In (Rhodes & Honkala, 1999), an implementation of HB analysis on shared memory multicomputers has been reported, where the parallel task allocation and scheduling are applied to device model evaluation, matrix-vector products and the standard block-diagonal (BD) preconditioner (Feldmann et al., 1996). In the literature, parallel matrix computation and parallel fast fourier transform / inverse fast fourier transform (FFT/IFFT) have also been exploited for harmonic balance. Some examples of the above ideas can be found from (Basermann et al., 2005; Mayaram et al., 1990; Sosonkinaet al., 1998).
In this chapter, we present a parallel approach that focuses on a key component of modern harmonic balance simulation engines, the preconditioner. The need in solving large practical harmonic balance problems has promoted the use of efficient iterative numerical methods, such as GMRES (Feldmann et al., 1996; Saad, 2003), and hence the preconditioning techniques associated with iterative methods. Under such context, preconditioning is a key as it not only determines the efficiency and robustness of the simulation, but also corresponds to a fairly significant portion of the overall compute work. The presented work is based upon a custom hierarchical harmonic balance preconditioner that is tailored to have improved efficiency and

robustness, and parallelizable by construction (Dong & Li, 2007a;b; 2009a; Li & Pileggi, 2004). The latter stems from the fact that the top-level linearized HB problem is decomposed into a series of smaller independent matrix problems across multiple levels, resulting a tree-like data dependency structure. This naturally provides a coarse-grained parallelization opportunity as demonstrated in this chapter.

In contrast to the widely used standard block-diagonal (BD) preconditioning (Feldmann et al., 1996; Rhodes & Honkala, 1999), the presented approach has several advantages First, purely from an algorithmic point of view, the hierarchical preconditioner possess noticeably improved efficiency and robustness, especially for strongly nonlinear harmonic balance problems (Dong & Li, 2007b; Li & Pileggi, 2004) . Second, from a computational point of view, the use of the hierarchical preconditioner pushes more computational work onto preconditioning, making an efficient parallel implementation of the preconditioner more appealing. Finally, the tree-like data dependency of the presented preconditioner allows for nature parallelization; in addition, freedoms exist in terms of how the overall workload corresponding to this tree may be distributed across multiple processors or compute nodes with a suitable granularity to suit a specific parallel computing platform.

The same core parallel preconditioning technique can be applied to not only standard steady-state analysis of driven circuits, but also that of autonomous circuits such as oscillators. Furthermore, it can be used as a basis for developing harmonic-balance based envelope-following analysis, critical to communication applications. This leads to a unifying parallel simulation framework targeting a range of steady-state and envelope following analyses. This framework also admits traditional parallel ideas that are based upon parallel evaluations of device models, parallel FFT/IFFT operations, and finer grained matrix-vector products. We demonstrate favorable runtime speedups that result from this algorithmic change, through the adoption of the presented preconditioner as well as parallel implementation, on computer clusters using message-passing interface (MPI) (Dong & Li, 2009a). Similar parallel runtime performances have been observed on multi-core shared-memory platforms.

2. Harmonic balance

A circuit with n unknowns can be described using the standard modified nodal analysis (MNA) formulation (Kundert et al., 1990)

$$h(t) = \frac{d}{dt}q(x(t)) + f(x(t)) - u(t) = 0, \tag{1}$$

where $x(t) \in \Re^n$ denotes the vector of n unknowns, $q(x(t)) \in \Re^n$ represents the vector of the charges/fluxes contributed by dynamic elements, $f(x(t)) \in \Re^n$ represents the vector of the currents contributed by static elements, and $u(t)$ is the vector of the external input excitations. If N harmonics are used to represent the steady-state circuit response in the frequency domain, the HB system of the equations associated with Equation 1 can be formulated as

$$H(X) = \Omega\Gamma q(\cdot)\Gamma^{-1}X + \Gamma f(\cdot)\Gamma^{-1}X - U = 0, \tag{2}$$

where X is the Fourier coefficient vector of circuit unknowns; Ω is a diagonal matrix representing the frequency domain differentiation operator; Γ and Γ^{-1} are the N-point FFT and IFFT (inverse FFT) matrices; $q(\cdot)$ and $f(\cdot)$ are the time-domain charge/flux and resistive equations defined above; and U is the input excitation in the frequency domain. When

the double-sided FFT/IFFT are used, a total number of $N = 2k + 1$ frequency components are included to represent each signal, where k is the number of positive frequencies being considered.
It is customary to apply the Newton's method to solve the nonlinear system in Equation 2. At each Newton iteration, the Jacobian matrix $J = \partial H/\partial X$ needs to be computed, which is written in the following matrix form (Feldmann et al., 1996; Kundert et al., 1990)

$$J = \Omega\Gamma C\Gamma^{-1} + \Gamma G\Gamma^{-1}, \tag{3}$$

where $C = diag\{c_k = \frac{\partial q}{\partial x}|_{x=x(t_k)}\}$ and $G = diag\{g_k = \frac{\partial f}{\partial x}|_{x=x(t_k)}\}$ are block-diagonal matrices with the diagonal blocks representing the linearizations of $q(\cdot)$ and $f(\cdot)$ at N sampled time points $t_1, t_2, \cdots, t_N$. The above Jacobian matrix is rather dense. For large circuits, storing the whole Jacobian matrix explicitly can be expensive. This promotes the use of an iterative method, such as Generalized Minimal Residual (GMRES) method or its flexible variant (FGMRES) (Saad, 1993; 2003). In this case, the Jacobian matrix needs only to be constructed *implicitly*, leading to the notion of the matrix-free formulation. However, an effective preconditoner shall be applied in order to ensure efficiency and convergence. To this end, preconditioning becomes an essential component of large-scale harmonic balance analysis.
The widely-used BD preconditioner discards the off-diagonal blocks in the Jacobian matrix by averaging the circuit linearizations at all discretized time points and uses the resulting block-diagonal approximation as a preconditioner (Feldmann et al., 1996). This relatively straightforward approach is effective for mildly nonlinear circuits, where off-diagonal blocks in the Jacobian matrix are not dominant. However, the performance of the BD preconditoner deteriorates as circuit nonlinearities increase. In certain cases, divergence may be resulted for strongly nonlinear circuits.

3. Parallel hierarchical preconditioning

A basic analysis flow for harmonic analysis is shown in Fig.1.
Clearly, at each Newton iteration, device model evaluation and the solution of a linearized HB problem must be performed. Device model evaluation can be parallelized easily due its apparent data-independent nature. For the latter, matrix-vector products and preconditioning are the two key operations. The needed matrix-vector products associated with Jacobian matrix J in Equation 3 are in the form

$$JX = \Omega(\Gamma(C(\Gamma^{-1}X))) + \Gamma(G(\Gamma^{-1}X)), \tag{4}$$

where G, C, Ω, Γ are defined in Section 2. Here, FFT/IFFT operations are applied independently to different signals, and hence can be straightforwardly parallelized. For preconditioning, we present a hierarchical scheme with improved efficiency and robustness, which is also parallelizable by construction.

3.1 Hierarchical harmonic balance preconditioner

To construct a parallel preconditioner to solve the linearized problem $JX = B$ defined by Equation 4, we shall identify the parallelizable operations that are involved. To utilize, say m,

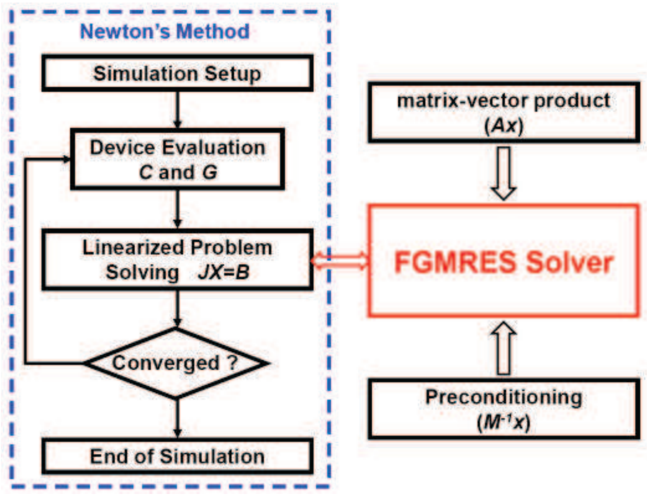

Newton's Method
Simulation Setup
Device Evaluation C and G
Linearized Problem Solving JX=B
Converged ?
End of Simulation
matrix-vector product (Ax)
FGMRES Solver
Preconditioning ($M^{-1}x$)

Fig. 1. A basic flow for HB analysis (from (Dong & Li, 2009a) ©[2009] IEEE).

processing elements (PEs), we rewrite Equation 4 as

$$\begin{bmatrix} J_{11} & J_{12} & \cdots & J_{1m} \\ J_{21} & J_{22} & \cdots & J_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ J_{m1} & J_{m2} & \cdots & J_{mm} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_m \end{bmatrix} = \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_m \end{bmatrix}, \tag{5}$$

where Jacobian J is composed of $m \times m$ block entries; X and B are correspondingly partitioned into m segments along the frequency boundaries. Further, J can be expressed in the form

$$[J]_{m\times m} = \left(\begin{bmatrix} \Omega_1 & & & \\ & \Omega_2 & & \\ & & \ddots & \\ & & & \Omega_m \end{bmatrix} C_c + G_c \right), \tag{6}$$

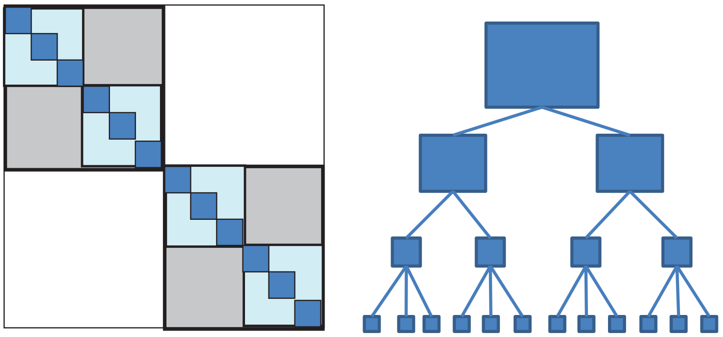
where circulants C_c, G_c are correspondingly partitioned as

$$\begin{aligned} C_c = \Gamma C \Gamma^{-1} &= \begin{bmatrix} C_{c11} & \cdots & C_{c1m} \\ \vdots & \ddots & \vdots \\ C_{cm1} & \cdots & C_{cmm} \end{bmatrix} \\ G_c = \Gamma G \Gamma^{-1} &= \begin{bmatrix} G_{c11} & \cdots & G_{c1m} \\ \vdots & \ddots & \vdots \\ G_{cm1} & \cdots & G_{cmm} \end{bmatrix} \end{aligned}. \tag{7}$$

A parallel preconditioner is essentially equivalent to a parallelizable approximation to J. Assuming that the preconditioner is going to be parallelized using m PEs, we discard the off-diagonal blocks of Equation 7, leading to m decoupled linearized problems of smaller dimensions

$$\begin{cases} J_{11}X_1 = [\Omega_1 C_{c11} + G_{c11}]X_1 = B_1 \\ J_{22}X_2 = [\Omega_2 C_{c22} + G_{c22}]X_2 = B_2 \\ \vdots \\ J_{mm}X_m = [\Omega_m C_{cmm} + G_{cmm}]X_m = B_m \end{cases} . \tag{8}$$

By solving these decoupled linearized problems in a parallel way, a parallel preconditioner is efficiently provided.



(a) Matrix view (b) Task dependence view

Fig. 2. Hierarchical harmonic balance preconditioner.

This basic idea of divide-and-conquer can be extended in a hierarchical fashion as shown in Fig. 2. At the topmost level, to solve the top-level linearized HB problem, a preconditioner is created by approximating the full Jacobian using a number (in this case two) of *super* diagonal blocks. Note that the partitioning of the full Jacobian is along the frequency boundary. That is, each matrix block corresponds to a selected set of frequency components of all circuit nodes in the fashion of Equation 5. These super blocks can be large in size such that an iterative method such as FGMRES is again applied to each such block with a preconditioner. These lower-level preconditioners are created in the same fashion as that of the top-level problem by recursively decomposing a large block into smaller ones until the block size is sufficiently small for direct solve.

Another issue that deserves discussion is the storage of each subproblem in the preconditioner hierarchy. Note that some of these submatrix problems are large. Therefore, it is desirable to adopt the same implicit matrix-free presentation for subproblems. To achieve this, it is critical to represent each linearized sub-HB problem using a sparse time-domain representation, which has a decreasing time resolution towards the bottom of the hierarchy consistent with the size of the problem. An elegant solution to this need has been presented in (Dong & Li, 2007b; Li & Pileggi, 2004), where the top-level time-varying linearizations of device characteristics are successively low-pass filtered to create time-domain waveforms with decreasing resolution for the sub-HB problems. Interested readers are redirected to (Dong & Li, 2007b; Li & Pileggi, 2004) for an in-depth discussion.

3.2 Advantages of the hierarchical preconditioner

Purely from a numerical point of view, the hierarchical preconditioner is more advantageous over the standard BD preconditioner. It provides a better approximation to the Jacobian, hence leading to improved efficiency and robustness, especially for strongly nonlinear circuits.
Additionally, it is apparent from Fig. 2 that there exists inherent data independence in the hierarchical preconditioner. All the subproblems at a particular level are fully independent, allowing natural parallelization. The hierarchial nature of the preconditioner also provides additional freedom and optimization in terms of parallelization granularity, and workload distribution, and tradeoffs between parallel efficiency and numerical efficiency. For example, the number of levels and the number of subproblems at each level can be tuned for the best runtime performance and optimized to fit a specific a parallel hardware system with a certain number of PEs. In addition, difference in processing power among the PE's can be also considered in workload partitioning, which is determined by the construction of the tree-like hierarchical structure of the preconditioner.

4. Runtime complexity and parallel efficiency

Different configurations of the hierarchial preconditioner lead to varying runtime complexities and parallel efficiencies. Understanding the tradeoffs involved is instrumental for optimizing the overall efficiency of harmonic balance analysis.
Denote the number of harmonics by M, the number of circuit nodes by N, the number of levels in the hierarchical preconditioner by K, the total number of sub-problems at level i by P_i ($P_1 = 1$ for the topmost level), and the maximum number of FGMRES iterations required to reach the convergence for a sub-problem at level i by $I_{F,i}$. We further define $S_{F,i} = \Pi_{k=1}^{i} I_{F,k}$, $i = 1, \cdots, K$ and $S_{F,0} = 1$.
The runtime cost in solving a sub-problem at the ith level can be broken into two parts: c1) the cost incurred by the FGMRES algorithm; and c2) the cost due to the preconditioning. In the serial implementation, the cost c1 at the topmost level is given by: $\alpha I_{F,1} MN + \beta I_{F,1} MN \log M$, where α, β are certain constants. The first term in c1 corresponds to the cost incurred within the FGMRES solver and it is assumed that a restarted (F)GMRES method is used. The second term in c1 represents the cost of FFT/IFFT operations. At the topmost level, the cost c2 comes from solving P_2 sub-problems at the second level $I_{F,1}$ times, which is further equal to the cost of solving all the sub-problems starting from the second level in the hierarchial preconditioner. Adding everything together, the total computational complexity of the serial hierarchically-preconditioned HB is

$$MN \sum_{i=1}^{K-1} P_i S_{F,i-1} \left(\alpha + \beta \log \frac{M}{P_i} \right) + \gamma S_{F,K} MN^{1.1}, \tag{9}$$

where the last term is due to the direct solve of the diagonal blocks of size N at the bottom of the hierarchy. We have assumed that directly solving an $N \times N$ sparse matrix problem has a cost of $O(N^{1.1})$.
For the parallel implementation, we assume that the workload is evenly split among m PEs and the total inter-PE communication overhead is T_{comm}, which is proportional to the number of inter-PE communications. Correspondingly, the runtime cost for the parallel implementation is

$$\frac{MN \sum_{i=1}^{K-1} P_i S_{F,i-1} \left(\alpha + \beta \log \frac{M}{P_i} \right) + \gamma S_{F,K} MN^{1.1}}{m} + T_{comm}. \tag{10}$$

It can be seen that minimizing the inter-PE communication overhead (T_{comm}) is important in order to achieve a good parallel processing efficiency factor. The proposed hierarchical preconditioner is parallelized by simultaneously computing large chunks of independent computing tasks on multiple processing elements. The coarse-grain nature of our parallel preconditioner reduces the relative contribution of the inter-PE communication overhead and contributes to good parallel processing efficiency.

5. Workload distribution and parallel implementation

We discuss important considerations in distributing the work load across multiple processing elements and parallel implementation.

5.1 Allocation of processing elements

We present a more detailed view of the tree-like task dependency of the hierarchical preconditioner in Fig. 3.

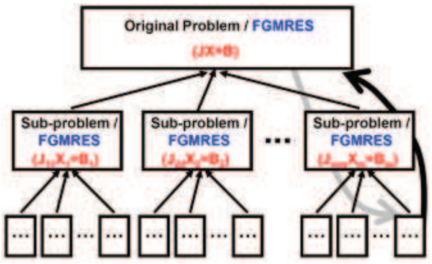


Fig. 3. The task-dependency graph of the hierarchical preconditioner (from (Dong & Li, 2009a) ©[2009] IEEE) .

5.1.1 Allocation of homogenous PE's

For PE allocation, let us first consider the simple case where the PEs are identical in compute power. Accordingly, each (sub)problem in the hierarchical preconditioner is split into N equally-sized sub-problems at the next level and the resulting sub-problems are assigned to different PE's. We more formally consider the PE allocation problem as the one that assigns a set of P PEs to a certain number of computing tasks so that the workload is balanced and there is no deadlock. We use the breadth-first traversal of the task dependency tree to allocate PEs, as shown in Algorithm 1.

The complete PE assignment is generated by calling $Allocate(root, P_{all})$, where the *root* is the node corresponding to the topmost linearized HB problem, which needs to be solved at each Newton iteration. P_{all} is the full set of PEs. We show two examples of PE allocation in Fig. 4 for the cases of three and nine PEs, respectively. In the first case, three PEs are all utilized at the topmost level. From the second level and downwards, a PE is only assigned to solve a sub-matrix problem and its children problems. Similarly, in the latter case, the workload at the topmost level is split between nine PEs. The difference from the previous case is that there are less number of subproblems at the second level than that of available PEs. These three subproblems are solved by three groups of PEs: $\{P_1, P_2, P_3\}$, $\{P_4, P_5, P_6\}$ and $\{P_7, P_8, P_9\}$, respectively. On the third level, a PE is assigned to one child problem of the corresponding parent problem at the second level.

Algorithm 1 Homogenous PE allocation

Inputs: a problem tree with root **n**; a set of **P** PEs with equal compute power;
Each problem is split into **N** sub-problems at the next level;
Allocate(n, P)

```
 1: Assign all PEs from $P$ to root node
 2: If $n$ does not have any child, return
 3: Else
 4:   Partition $P$ into $N$ non-overlapping subsets, $P^1, P^2, \cdots, P^N$:
 5:     IF $\lfloor \frac{P}{N} \rfloor == \frac{P}{N}$
 6:       $P^i$ has $P/N$ PEs $(1 \leq i \leq N)$
 7:     Elseif $(P > N)$
 8:       $P^i$ has $\lfloor \frac{P}{N} \rfloor + 1$ PEs $(1 \leq i < N)$ and
          $P^N$ has $P - (\lfloor \frac{P}{N} \rfloor + 1)(N-1)$ PEs
 9:     Else
10:       $P^i$ has one PE $(1 \leq i \leq P)$ and others have no PE; return a warning message
11:   For each child $n_i$: Allocate($n_i$, $P^i$).
```

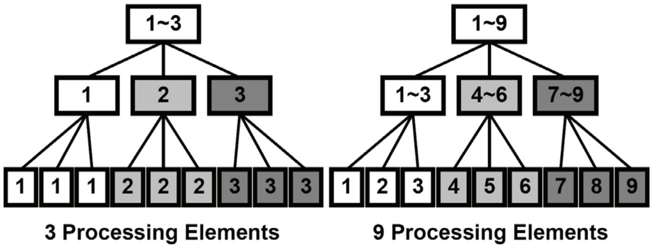


Fig. 4. Examples of homogenous PE allocation (from (Dong & Li, 2009a) ©[2009] IEEE).

5.1.2 Deadlock avoidance

A critical issue in parallel processing is the avoidance of deadlocks. As described as follows, deadlocks can be easily avoided in the PE assignment. In general, a deadlock is a situation where two or more dependent operations wait for each other to finish in order to proceed. In an MPI program, a deadlock may occur in a variety of situations (Vetter et al., 2000). Let us consider Algorithm 1. PEs P_1 and P_2 are assigned to solve matrix problems M_A and M_B on the same level. Naturally, P_1 and P_2 may be also assigned to solve the sub-problems of M_A and M_B, respectively. Instead of this, if one assigns P_1 to solve a sub-problem of M_B and P_2 a sub-problem of M_A, a deadlock may happen. To make progress on both solves, the two PEs may need to send data to each other. When P_1 and P_2 simultaneously send the data and the system does not have enough buffer space for both, a deadlock may occur. It would be even worse if several pairs of such operations happen at the same time. The use of Algorithm 1 reduces the amount of inter-PE data transfer, therefore, avoids certain deadlock risks.

5.1.3 Allocation of heterogenous PE's

It is possible that a parallel system consists of processing elements with varying compute power. Heterogeneity among PEs can be considered in the allocation to further optimize the performance. In this situation, subproblems with different sizes may be assigned to each PE. We show a size-dependent allocation algorithm in Algorithm 2. For ease of presentation, we have assumed that the runtime cost of linear matrix solves is linear in problem size. In practice, more accurate runtime estimates can be adopted.

Algorithm 2 Size-dependent Heterogenous PE allocation

```
Inputs: a problem tree with root n; a set of P PEs; problem size S;
each problem is split into N sub-problems at the next level;
compute powers are represented using weights of PEs : $w_1 \leq w_2 \leq \cdots \leq w_P$
Allocate(n, P, S)
1: Assign all PEs to root node
2: If $n$ does not have any child, return
3: Else
4:    Partition $P$ into $N$ non-overlapping subsets: $P^1, P^2, \cdots, P^N$,
      with the total subset weights $w_{s,i}$, $(1 \leq i \leq N)$.
5:    Minimize the differences between $w_{s,i}$'s.
6:    Choose the size of the i-th child node $n_i$ as:
      $S_i = S \cdot w_{s,i} / \sum_{j=1}^{P} w_j$
7:    For each $n_i$: Allocate($n_i$, $P^i$, $S_i$ ).
```

An illustrative example is shown in Fig. 5. Each problem is recursively split to three sub-problems at the next level. The subproblems across the entire tree are denoted by $n_i, (1 \leq i \leq 13)$. These problems are mapped onto nine PEs with compute power weights $w_1 = 9$, $w_2 = 8$, $w_3 = 7$, $w_4 = 6$, $w_5 = 5$, $w_6 = 4$, $w_7 = 3$, $w_8 = 2$ and $w_9 = 1$, respectively. According to Algorithm 2, we first assign all PEs ($P_1 \sim P_9$) to n_1, the top-level problem. At the second level, we cluster the nine PEs to three groups and map a group to a sub-problem at the second level. While doing this, we minimize differences in total compute power between these three groups. We assign $\{P_1, P_6, P_7\}$ to n_2, $\{P_2, P_5, P_8\}$ to n_3, and $\{P_3, P_4, P_9\}$ to n_4, as shown in Fig. 5. The sum of compute power of all the PE's is 45, while those allocated to n_2, n_3 and n_4 are 16, 15 and 14, respectively, resulting a close match. A similar strategy is applied at the third-level of the hierarchical preconditioner as shown in Fig. 5.

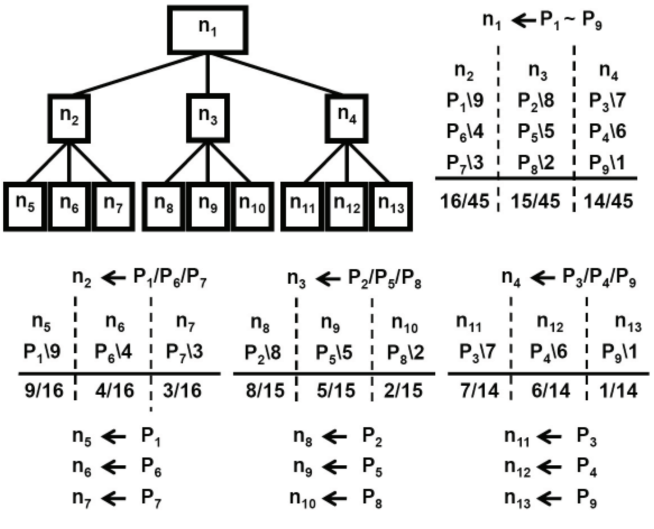


Fig. 5. Example of size-dependent heterogenous PE allocation (from (Dong & Li, 2009a) ©[2009] IEEE).

5.2 Parallel implementation

The proposed parallel preconditioner can be implemented in a relatively straightforward way either on distributed platforms using MPI or on shared-memory platforms using pThreads due to its coarse grain nature. Both implementations have been taken and comparisons were made between the two. Similar parallel scaling characteristics for both implementations have been observed, again, potentially due to the coarse grain nature of the proposed preconditioner.

We focus on some detailed considerations for the MPI based implementation. On distributed platforms, main parallel overheads come from inter-PE communications over the network. Therefore, one main implementation objective is to reduce the communication overhead among the networked workstations. For this purpose, non-blocking MPI routines are adopted instead of their blocking counterparts to overlap computation and communication. This strategy entails certain programming level optimizations.

As an example, consider the situation depicted in Fig. 5. The solutions of subproblems n_5, n_6 and n_7 computed by PEs P_1, P_6 and P_7, respectively, need to be all sent to one PE, say P_1, which also works on a higher-level parent problem. Since multiple sub-problems are being solved concurrently, P_1 may not immediately respond to the requests from P_6 (or P_7). This immediately incurs performance overhead if blocking operations are used.

Instead, one may adopt non-blocking operations, as shown in Fig. 6, where a single data transfer is split into several segments. At a time, P_6 (or P_7) only prepares one segment of data and sends a request to P_1. Then, the PE can prepare the next segment of data to be sent. As such, the communication and computation can be partially overlapped.

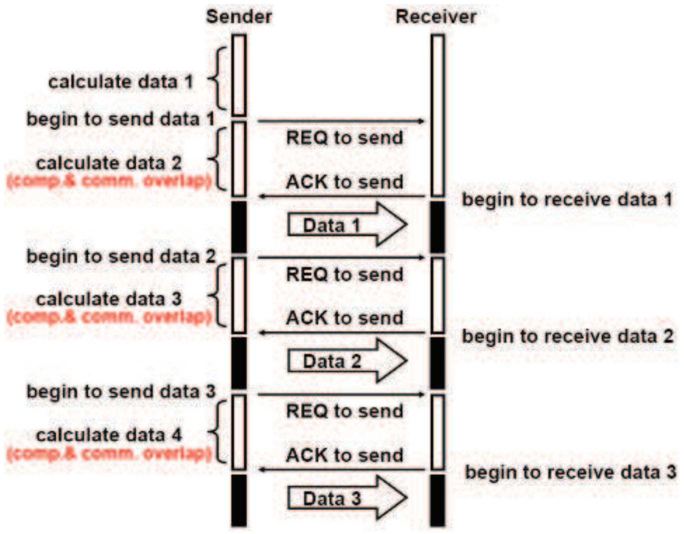


Fig. 6. Alleviating communication overhead via non-blocking data transfers (from (Dong & Li, 2009a) ©[2009] IEEE).

Note that the popularity of recent multi-core processors has stimulated the development of multithreading based parallel applications. Inter-PE communication overheads may be reduced on shared-memory multi-core processors. This may be particularly beneficial for fine grained parallel applications. In terms of parallel circuit simulation, for large circuits, issues resulted from limited shared-memory resources must be carefully handled.

6. Parallel autonomous circuit and envelope-following analyses

Under the context of driven circuits, we have presented the hierarchical preconditioning technique in previous sections. We further show that the same approach can be extended to harmonic balance based autonomous circuit steady-state and envelope-following analyses.

6.1 Steady-state analysis of autonomous circuits

Several simulation techniques have been developed for the simulation of autonomous circuits such as oscillators (Boianapally et al., 2005; Duan & Mayaram, 2005; Gourary et al., 1998; Kundert et al., 1990; Ngoya et al., 1995). In the two-tier approach proposed in (Ngoya et al., 1995), the concept of voltage probe is introduced to transform the original autonomous circuit problem to a set of closely-related driven circuit problems for improved efficiency. As shown in Fig. 7, based on some initial guesses of the probe voltage and the steady-state frequency, a driven-circuit-like HB problem is formulated and solved at the second level (the lower tier). Then, the obtained probe current is used to update the probe voltage and the steady-state frequency at the top level (the upper tier). The process repeats until the probe current becomes (approximately) zero.

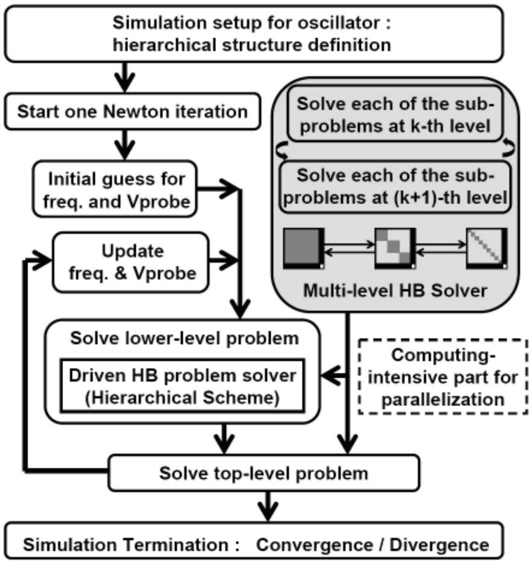

Simulation setup for oscillator : hierarchical structure definition
Start one Newton iteration
Initial guess for freq. and Vprobe
Update freq. & Vprobe
Solve each of the sub-problems at k-th level
Solve each of the sub-problems at (k+1)-th level
Multi-level HB Solver
Solve lower-level problem
Driven HB problem solver (Hierarchical Scheme)
Computing-intensive part for parallelization
Solve top-level problem
Simulation Termination : Convergence / Divergence

Fig. 7. Parallel harmonic balance based autonomous circuit analysis (from (Dong & Li, 2009a) ©[2009] IEEE).

It is shown as follows that the dominant cost of this two-tier approach comes from a series of analysis problems whose structure resembles that of a driven harmonic balance problem, making it possible to extend the aforementioned hierarchical preconditioner for analyzing oscillators.

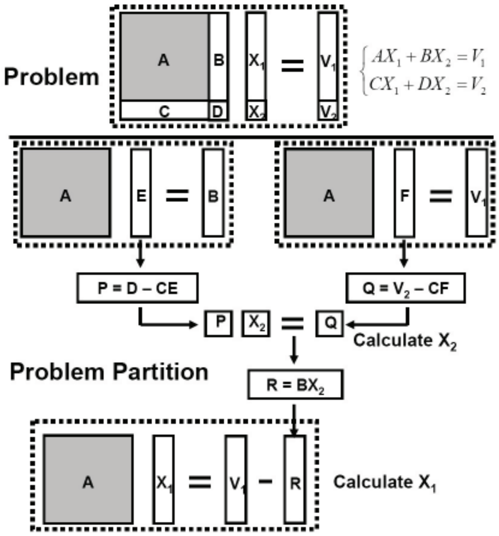


Fig. 8. Partitioning of the Jacobian of autonomous circuits (from (Dong & Li, 2009a) ©[2009] IEEE).

In the two-tier approach, the solution of the second-level HB problem dominates the overall computational complexity. We discuss how these second level problems can be sped up by an extended parallelizable hierarchical preconditioner. The linearized HB problem at the lower tier corresponds to an extended Jacobian matrix

$$\begin{bmatrix} A_{nN\times nN} & B_{nN\times l} \\ C_{l\times nN} & D_{l\times l} \end{bmatrix} \cdot X_{(nN+l)\times 1} = V_{(nN+l)\times 1}, \tag{11}$$

where n and N are the numbers of the circuit unknowns and harmonics, respectively, and $l(l << nN)$ is the number of additionally appended variables corresponding to the steady-state frequency and the probe voltage. It is not difficult to see that the structure of matrix block $A_{nN\times nN}$ is identical to the Jacobian matrix of a driven circuit HB analysis. Equation 11 is rewritten in the following partitioned form

$$\begin{cases} AX_1 + BX_2 = V_1 \\ CX_1 + DX_2 = V_2 \end{cases}. \tag{12}$$

From the first equation in Equation 12, we express X_1 in terms of X_2 as

$$X_1 = A^{-1}(V_1 - BX_2). \tag{13}$$

Substituting Equation 13 into the second equation in Equation 12 gives

$$X_2 = (D - CA^{-1}B)^{-1}(V_2 - CA^{-1}V_1). \tag{14}$$

The dominant computational cost for getting X_2 comes from solving the two linearized matrix problems associated with $A^{-1}B$ and $A^{-1}V_1$. When X_2 is available, X_1 can be obtained by solving the third matrix problem defined by A in Equation 13, as illustrated in Fig. 8.
Clearly, the matrix structure of these three problems is defined by matrix A, which has a structure identical to the Jacobian of a driven circuit. The same hierarchical preconditioning idea can be applied to accelerate the solutions of the three problems.

6.2 Envelope-following analysis

Envelope-following analysis is instrumental for many communication circuits. It is specifically suitable for analyzing periodic or quasi-periodic circuit responses with slowly varying amplitudes (Feldmann& Roychowdhury, 1996; Kundert et al., 1988; Rizzoli et al., 1999; 2001; Silveira et al., 1991; White & Leeb, 1991). The principal idea of the HB-based envelope-following analysis is to handle the slowly varying amplitude, called envelope, of the fast carrier separately from the carrier itself, which requires the following mathematical representation of each signal in the circuit

$$x(t) = \sum_{k=-K}^{K} X_k(t) e^{jk\omega_0 t}, N = 2K+1, \tag{15}$$

where the envelope $X_k(t)$ varies slowly with respect to the period of the carrier $T_0 = 2\pi/\omega_0$. This signal representation is illustrated in Fig. 9.
As a result, the general circuit equation in Equation 1 can be cast to

$$h(t) = h(t_e, t_c) = \sum_{k=-K}^{K} [jk\omega_0 Q_k(t_e) + \frac{d}{dt} Q_k(t_e) + G_k(t_e) - U_k(t_e)] e^{jk\omega_0 t_c}, \tag{16}$$

where different time variables t_e, t_c are used for the envelope and the carrier. Correspondingly, the Fourier coefficients shall satisfy

$$H(X(t_e)) = \Omega\Gamma q(\cdot)\Gamma^{-1}X(t_e) + \frac{d}{dt_e}\Gamma q(\cdot)\Gamma^{-1}X(t_e) + \Gamma f(\cdot)\Gamma^{-1}X(t_e) - U(t_e) = 0, \tag{17}$$

which can be solved by using a numerical integration method. Applying Backward Euler (BE) to discretize Equation 17 over a set of time points $(t_1,\ t_2,\ \cdots,\ t_q,\ \cdots)$ leads to

$$\begin{aligned} &\left(\Gamma q(\cdot)\Gamma^{-1}X(t_q) - \Gamma q(\cdot)\Gamma^{-1}X(t_{q-1})\right) / (t_q - t_{q-1}) \\ &+\Omega\Gamma q(\cdot)\Gamma^{-1}X(t_q) + \Gamma f(\cdot)\Gamma^{-1}X(t_q) - U(t_q) = 0. \end{aligned} \tag{18}$$

To solve this nonlinear problem using the Newton's method, the Jacobian is needed

$$\begin{aligned} &J_{env} = \tfrac{\Gamma C\Gamma^{-1}}{t_q - t_{q-1}} + \Omega\Gamma C\Gamma^{-1} + \Gamma G\Gamma^{-1} = \\ &\begin{bmatrix} \Omega_1 + \frac{I_1}{t_q - t_{q-1}} & & \\ & \ddots & \\ & & \Omega_m + \frac{I_m}{t_q - t_{q-1}} \end{bmatrix} \cdot C_c + G_c, \end{aligned} \tag{19}$$

where the equation is partitioned into m blocks in a way similar to Equation 6; $I_1,\ I_2,\ \cdots,\ I_m$ are identity matrices with the same dimensions as the matrices $\Omega_1, \Omega_2, \cdots, \Omega_m$, respectively; Circulants C_c and G_c have the same forms as in Equation 7. Similar to the treatment taken in Equation 8, a parallel preconditioner can be formed by discarding the off-block diagonal entries of Equation 7, which leads to m decoupled linear problems of smaller dimensions

$$\begin{cases} [(\Omega_1 + \frac{I_1}{(t_q - t_{q-1})})C_{c11} + G_{c11}]X_1 = B_1 \\ [(\Omega_2 + \frac{I_2}{(t_q - t_{q-1})})C_{c22} + G_{c22}]X_2 = B_2 \\ \vdots \\ [(\Omega_m + \frac{I_m}{(t_q - t_{q-1})})C_{cmm} + G_{cmm}]X_m = B_m \end{cases} . \tag{20}$$

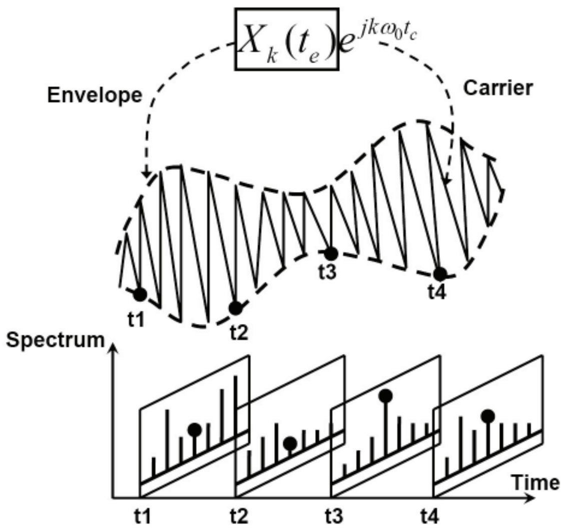


Fig. 9. Signal representations in envelope-following analysis (from (Dong & Li, 2009a) ©[2009] IEEE).

To summarize, the mathematical structure of these sub-problems is identical to that of a standard HB problem. The same matrix-free representation can be adopted to implicitly form these matrices. A hierarchical preconditioner can be constructed by applying the above decomposition recursively as before.

7. Illustrative examples

We demonstrate the presented approach using a C/C++ based implementation. The MPICH library (Gropp & Lusk, 1996) has been used to distribute the workload over a set of networked Linux workstations with a total number of nine CPUs. The FFTW package is used for FFT/IFFT operations (Frigo & Johnson, 2005) and the FGMRES solver is provided through the PETSC package (Balay et al., 1996). Most of the parallel simulation results are based upon the MPI based implementation unless stated otherwise.

7.1 Simulation of driven circuits

A list of circuits in Table 1 are used in the experimental study. For the hierarchical preconditioning technique, a three-level hierarchy is adopted, where the size of each sub-problem is reduced by a factor of three at the next lower level.
Serial and parallel implementations of the block diagonal (BD) preconditioner (Feldmann et al., 1996) and the hierarchical preconditioner are compared in Table 2. Here a parallel implementation not only parallelizes the preconditioner, but also other parallelizable components such as device model evaluation and matrix-vector products. The second and third columns show the runtimes of harmonic balance simulations using the serial BD and hierarchical preconditioner, respectively. The columns below ’T3(s)’, ’T5(s)’ and ’T4(s)’, ’T6(s)’ correspond to the runtimes of the parallel HB simulations using the BD preconditioner and the hierarchical preconditioner, respectively. The columns below ’X1’-’X4’ indicate the parallel runtime speedups over the serial counterparts. It is clear that the hierarchical preconditioner

Index	Description of circuits	Nodes	Freqs	Unknowns
1	frequency divider	17	100	3,383
2	DC-DC converter	8	150	2,392
3	diode rectifier	5	200	1,995
4	double-balanced mixer	27	188	10,125
5	low noise amplifier	43	61	5,203
6	LNA + mixer	69	86	11,799
7	RLC mesh circuit	1,735	10	32,965
8	digital counter	86	50	8,514

Table 1. Descriptions of the driven circuits (from (Dong & Li, 2009a) ©[2009] IEEE).

speeds up harmonic balance simulation noticeably in the serial implementation. The MPI-based parallel implementation brings in additional runtime speedups.

Index	Serial		Parallel 3-CPU Platform				Parallel 9-CPU Platform			
	BD	Hierarchical	BD		Hierarchical		BD		Hierarchical	
	T1(s)	T2(s)	T3(s)	X1	T4(s)	X2	T5(s)	X3	T6(s)	X4
1	354	167	189	1.87	92	1.82	89	3.97	44	3.79
2	737	152	391	1.88	83	1.83	187	3.94	40	3.80
3	192	39	105	1.82	22	1.77	52	3.69	11	3.54
4	55	15	31	1.77	9	1.67	14	3.93	4	3.75
5	1,105	127	570	1.93	69	1.84	295	3.74	36	3.53
6	139	39	80	1.73	23	1.67	38	3.66	11	3.55
7	286	69	154	1.85	38	1.80	76	3.76	19	3.62
8	2,028	783	1,038	1.95	413	1.89	512	3.96	204	3.83

Table 2. Comparison on serial and parallel implementations of the two preconditioners (modified from (Dong & Li, 2009a) ©[2009] IEEE).

To show the parallel runtime scaling of the hierarchical preconditioner, the runtime speedups of the parallel preconditioner over its serial counterpart as a function of the number of processors for three test circuits are shown in Fig. 10.
In Fig. 11, we compare the distributed-memory based implementation using MPI with the shared-memory based implementation using multithreading (pThreads) for the frequency divider and the DC-DC converter. Two implementations exhibit a similar scaling characteristic. This is partially due to the fact the amount of inter-PE communication is rather limited in the proposed hierarchal preconidtioner. As a result, the potentially greater communication overhead of the distributed implementation has a limited impact on the overall runtimes.

7.2 Parallel simulation of oscillators

A set of oscillators described in Table 3 are used to compare two implementations of the two-tier method (Ngoya et al., 1995), one with the block-diagonal (BD) preconditioner, and the other the hierarchial preconditioner.
The runtimes of the serial implementations of the two versions are listed in the columns labeled as "Serial Platform" in Table 4. At the same time, the runtimes of the parallel simulations with the BD and hierarchical preconditioners on the 3-CPU and 9-CPU platforms are also shown in the table. The columns below ’X3’ and ’X5’ are the speedups of parallel simulations with the BD preconditioner. And the columns below ’X4’ and ’X6’ are the speedups of parallel simulations with the hierarchical preconditioner.

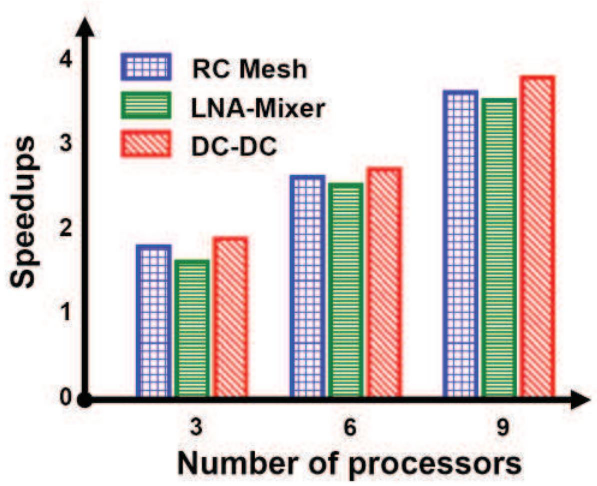


Fig. 10. The runtime speedups of harmonic balance simulation with hierarchical preconditioning as a function of the number of processors (from (Dong & Li, 2009a) ©[2009] IEEE).

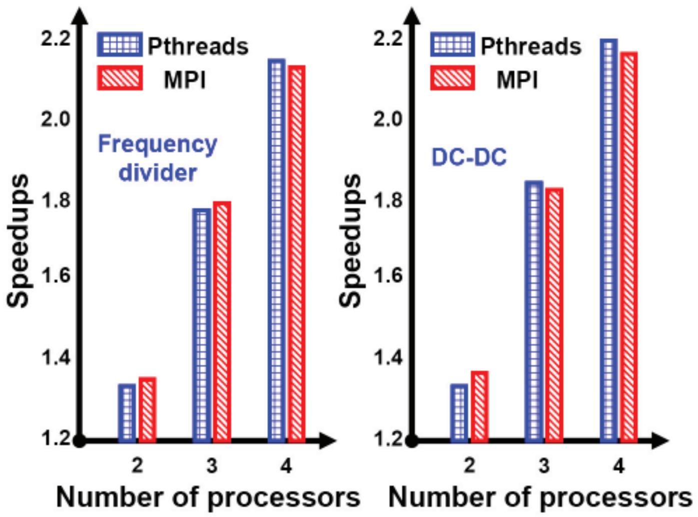


Fig. 11. Comparison of shared-memory and distributed-memory implementations of hierarchical preconditioning (from (Dong & Li, 2009a) ©[2009] IEEE).

Index	Oscillator	Nodes	Freqs	Unknowns
1	11 stages ring oscillator	13	50	1,289
2	13 stages ring oscillator	15	25	737
3	15 stages ring oscillator	17	20	665
4	LC oscillator	12	30	710
5	digital-controlled oscillator	152	10	2890

Table 3. Descriptions of the oscillators (from (Dong & Li, 2009a) ©[2009] IEEE).

Osc.	Serial Platform				Parallel 3-CPU Platform				Parallel 9-CPU Platform			
	Two-tier BD		Two-tier Hier.		BD		Hier.		BD		Hier.	
	T1(s)	N-Its	T2(s)	N-Its	T3(s)	X3	T4(s)	X4	T5(s)	X5	T6(s)	X6
1	127	48	69	43	74	1.71	41	1.68	32	3.97	18	3.83
2	95	31	50	27	55	1.73	29	1.72	24	3.96	13	3.85
3	83	27	44	23	48	1.73	26	1.69	22	3.77	12	3.67
4	113	42	61	38	67	1.68	37	1.66	30	3.80	17	3.69
5	973	38	542	36	553	1.76	313	1.73	246	3.95	141	3.86

Table 4. Comparisons of the two preconditioners on oscillators (from (Dong & Li, 2009a) ©[2009] IEEE).

On the 3-CPU platform, the average values below the columns ’X3’ and ’X4’ are 1.72x, 1.70x, respectively; On the 9-CPU platform, these average values are 3.89x and 3.78x respectively. It can be observed that the proposed parallel method brings favorable speedups over both its serial implementation and the parallel counterpart with the BD preconditioner.

7.3 Parallel envelope-following analysis

A power amplifier and a double-balanced mixer are used to demonstrate the proposed ideas, and the results are shown in Table 5. The runtimes are in seconds. As a reference, the runtimes of the serial transient simulation, the serial envelope-following simulations with the BD and the hierarchical preconditioners are listed in the columns below "Serial Platform", respectively. The columns below ’X2’ and ’X3’ indicate the speedups of the envelope-following simulation over the transient simulation. In the columns labeled as "3 CPUs" and "9 CPUs", the runtime results of the parallel envelope-following simulations with the BD preconditioner and the hierarchical preconditioner using three and nine CPUs are shown. The columns below ’X4’-’X7’ indicate the runtime speedups of the parallel envelope-following analyses over their serial counterparts. The runtime benefits of the proposed parallel approach are clearly seen.

CKT	Serial Platform					3 CPUs				9 CPUs			
	Trans.	BD		Hier.		BD		Hier.		BD		Hier.	
	T1	T2	X2	T3	X3	T4	X4	T5	X5	T6	X6	T7	X7
PA	831	76	10.9	26	32.0	44	1.73	16	1.64	19	4.01	7	3.72
Mixer	1,352	102	13.2	39	34.6	60	1.70	24	1.62	26	3.94	11	3.67

Table 5. Comparison of the two preconditioners on envelope-following simulation (from (Dong & Li, 2009a) ©[2009] IEEE).

8. Conclusions

We address the computational challenges associated with harmonic balance based analog and RF simulation from two synergistic angles: hierarchical preconditioning and parallel processing. From the first angle, we tackle a key computational component of modern harmonic balance algorithms that rely on the matrix-free implicit formulation and efficient iterative methods. The second angle is meaningful as parallel computing has become increasingly pervasive and utilizing parallel computing power is an effective means for improving the runtime efficiency of electronic design automation tools. The presented hierarchical preconditioner is numerically robust and efficient, and parallizable by construction. Favorable runtime performances of hierarchical preconditioning have been demonstrated on distributed and shared memory computing platforms for steady-state analysis of driven and automatous circuits as well as harmonic balance based envelope-following analysis.

9. Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 0747423, and SRC and Texas Analog Center for Excellence under contract 2008-HC-1836.

10. References

Balay, S.; Buschelman, K.; Gropp, W.; Kaushik, D.; Knepley, M.; McInnes, L.; Smith, B. & Zhang, H. (2001). *PETSc Web pages:* URL*: www.mcs.anl.gov/petsc*, Argonne National Laboratory.

Basermann, A.; Jaekel, U.; Nordhausen, M. & and Hachiya, K.(2005). Parallel iterative solvers for sparse linear systems in circuit simulation *Future Gener. Comput. Syst.*, Vol. 21, No. 8, (October 2005) pp. 1275-1284, ISSN 0167-739X.

Boianapally, K.; Mei, T. & Roychowdhury, J. (2005). A multi-harmonic probe technique for computing oscillator steady states, *Digest of Technical Papers, IEEE/ACM Int. Conf. on CAD*, pp. 610-613, ISBN 0-7803-9254-X, San Jose, CA, USA, November 2005, IEEE/ACM.

Dong, W. & Li, P. (2007a) Accelerating harmonic balance simulation using efficient parallelizable hierarchical preconditioning, *Proceedings of IEEE/ACM Design Automation Conference*, pp. 436-439, ISBN 978-1-59593-627-1, San Diego, CA, USA, June 2007, IEEE/ACM.

Dong, W.; & Li, P. (2007b). Hierarchical harmonic-balance methods for frequency-domain analog-circuit analysis, *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 26, No. 12, (December 2007) pp. 2089-2101, ISSN 0278-0070.

Dong, W., Li, P. & Ye, X. (2008) WavePipe: parallel transient simulation of analog and digital circuits on multi-core shared-memory machines, *Proceedings of IEEE/ACM Design Automation Conference*, pp. 238-243, ISBN 978-1-60558-115-6, Anaheim, CA, USA, June 2008, IEEE/ACM.

Dong, W. & Li, P. (2009a). A parallel harmonic balance approach to steady-state and envelope-following simulation of driven and autonomous circuits, *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 8, No. 4, (April 2009) pp. 490 - 501, ISSN 0278-0070.

Dong, W. & Li, P. (2009b) Parallelizable stable explicit numerical integration for efficient circuit simulation, *Proceedings of IEEE/ACM Design Automation Conference*, pp. 382-385, ISBN 978-1-6055-8497-3, San Francisco, CA, USA, July 2009, IEEE/ACM.

Dong, W.; & Li, P. (2009c). Final-value ODEs: stable numerical integration and its application to parallel circuit analysis, *Digest of Technical Papers, IEEE/ACM Int. Conf. on CAD*, pp. 73-78, ISBN 978-1-60558-800-1, San Jose, CA, USA, November 2009, IEEE/ACM.

Duan, X. & Mayaram, K. (2005). An efficient and robust method for ring-oscillator simulation using the harmonic-balance method, *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 24, No. 8, (August 2005) pp. 1225-1233, ISSN 0278-0070.

Feldmann, P.; Melville, R. & Long, D. (1996) Efficient frequency domain analysis of large nonlinear analog circuits, *Proceedings of IEEE Custom Integrated Circuts Conf.*, pp. 461-464, ISBN 0-7803-3117-6, San Diego, CA, USA, May 1996, IEEE.

Feldmann, P. & Roychowdhury, J. (1996). Computation of circuit waveform envelopes using an efficient matrix-decomposed harmonic balance algorithm, *Digest of Technical Papers, IEEE/ACM Int. Conf. on CAD*, pp. 295-300, ISBN 0-8186-7597-7, San Jose, CA, USA, November 1996, IEEE/ACM.

Frigo, M. & Johnson, S. (2005). The design and implementation of FFTW3, *Proceedings of the IEEE*, Vol. 93, No. 2, (January 2005) pp. 216-231, ISSN 0018-9219.

Gourary, M.; Ulyanov, S.; Zharov, M.; Rusakov, S.; Gullapalli, K.; & Mulvaney, B (1998). Simulation of high-Q oscillators, *Digest of Technical Papers, IEEE/ACM Int. Conf. on CAD*, pp. 162-169, ISBN 1-58113-008-2, San Jose, CA, USA, November 1998, IEEE/ACM.

Gropp, W. & Lusk, E. (1996). *User's Guide for* `mpich`*, a Portable Implementation of MPI*,Mathematics and Computer Science Division, Argonne National Laboratory.

Kundert, K.; White, J. & Sangiovanni-Vincentelli, A. (1988). An envelope-following method for the efficient transient simulation of switching power and filter circuits, *Digest of Technical Papers, IEEE/ACM Int. Conf. on CAD*, pp. 446-449, ISBN 0-8186-0869-2, San Jose, CA, USA, October 1988, IEEE/ACM.

Kundert, K.; White, J.; & Sangiovanni-Vincentelli, A. (1990). *Steady-state Methods for Simulating Analog and Microwave Circuits*, Kluwer Academic Publisher, ISBN 978-0-7923-9069-5, Boston, USA.

Li, P. & Pileggi, L.(2004). Efficient harmonic balance simulation using multi-level frequency decomposition, *Digest of Technical Papers, IEEE/ACM Int. Conf. on CAD*, pp. 677-682, ISBN 1092-3152, San Jose, CA, USA, November 2004, IEEE/ACM.

Mayaram, K.; Yang, P.; Burch, R. Chern, J.; Arledge, L. & Cox, P.(1990). A parallel block-diagonal preconditioned conjugate-gradient solution algorithm for circuit and device simulations, *Digest of Technical Papers, IEEE/ACM Int. Conf. on CAD*, pp. 446-449, ISBN 0-8186-2055-2, San Jose, CA, USA, November 1990, IEEE/ACM.

Ngoya, E.; Suarez, A.; Sommet, R. & Quere, R. (1995). Steady state analysis of free or forced oscillators by harmonic balance and stability investigation of periodic and quasi-periodic regimes, *nt. J. Microw. Millim.-Wave Comput.-Aided Eng.*, Vol. 5 No. 3, (March 1995) pp. 210-233, ISSN 1050-1827.

Reichelt, M.; Lumsdaine, A; & White, J. (1993). Accerlerated waveform methods for parallel transient simulation of semiconductor devices, *Digest of Technical Papers, IEEE/ACM Int. Conf. on CAD*, pp. 270–74, ISBN 0-8186-4490-7, San Jose, CA, USA, November 1993, IEEE/ACM.

Rhodes, D.; & Perlman, B. (1997). Parallel computation for microwave circuit simulation. *IEEE Trans. on Microwave Theory and Techniques*, Vol. 45, No. 5, (May 1997) pp. 587-592, ISSN 0018-9480.

Rhodes, D. & Gerasoulis, A.(1999). Scalable parallelization of harmonic balance simulation, *Proceedings of IPPS/SPDP Workshops*, pp. 1055-1064, ISBN 3-540-65831-9, San Juan, Puerto Rico, USA, April 1999, Springer.

Karanko, V.; & Honkala, M.(2004). A parallel harmonic balance simulator for shared memory multicomputers, *Proceedings of 34th European Microwave Conference*, pp. 849-851, ISBN 3-540-65831-9, Amsterdam, The Netherlands, October 2004, IEEE.

Rhodes, D. & Gerasoulis, A.(2000). A scheduling approach to parallel harmonic balance simulation. *Concurrency: Practice and Experience*, Vol. 12, No. 2-3, (February-March 2000) pp. 175-187.

Rizzoli, V.; Neri, A. Mastri, F. & and Lipparini, A. (1999). A krylov-subspace technique for the simulation of RF/microwave subsystems driven by digitally modulated carriers, *Int. J. RF Microwave Comput.-Aided Eng.*, Vol. 11, No. 7, (August 2005) pp. 490-505, ISSN 1531-1309.

Rizzoli, V.; Costanzo, A. & Mastri, F. (2001). Efficient krylov-subspace simulation of autonomous RF/microwave circuits driven by digitally modulated carriers, *IEEE Microwave Wireless Comp. Lett.*, Vol. 11, No. 7, (July 2001) pp. 308-310, ISSN 1531-1309.

Saad, Y. (1993). A flexible inner-outer preconditioned GMRES algorithm, *SIAM J.Sci.Comput.*, Vol. 14, No. 2, (March 1993) pp. 461-469, ISSN 1064-8275.

Lima, P.; Bonarini, A. & Mataric, M. (2003). *Iterative Methods for Sparse Linear Systems*, Society for Industrial and Applied Mathematics, ISBN 0898715342, Philadelphia, PA, USA.

Silveira, L.; White, J. & Leeb, S. (1991). A modified envelope-following approach to clocked analog circuit simulation, *Digest of Technical Papers, IEEE/ACM Int. Conf. on CAD*, pp. 20-23, ISBN 0-8186-2157-5, San Jose, CA, USA, November 1991, IEEE/ACM.

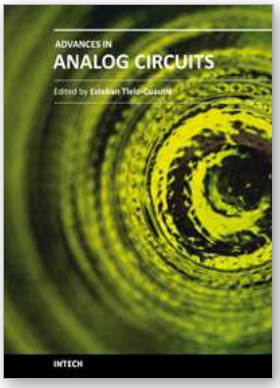
Sosonkina, M., Allison, D. & Watson, L. (1998). Scalable parallel implementations of the gmres algorithm via householder reflections, *Int. Conf. on Parallel Processing*, pp. 396-404, ISBN 0-8186-8650-2, Minneapolis, MN, USA, August 1998, IEEE.

Vetter, J.; & de Supinski, B. (2000) Dynamic software testing of MPI applications with Umpire, *Proceedings of the 2000 ACM/IEEE conference on Supercomputing*, pp. 51-60, ISBN 0-7803-9802-5, Washington, DC, USA, November 2000, IEEE.

Wever, U. & Zheng, Q. (1996) Parallel Transient Analysis for Circuit Simulation, *Proceedings of the Twenty-Ninth Hawaii Int. Conf. on System Sciences*, pp. 442-447, ISBN 0-8186-7324-9, Wailea, HI, USA, January 1996, IEEE.

White, J. & Leeb, S. (1991). An envelope-following approach to switching power converter simulation, *IEEE Trans. on Power Electronics*, Vol. 6, No. 2, (April 1991) pp. 303-307, ISSN 0885-8993.

Ye, X.; Dong, W.; Li, P. & Nassif, S. (2008). MAPS: Multi-Algorithm Parallel circuit Simulation, *Digest of Technical Papers, IEEE/ACM Int. Conf. on CAD*, pp. 73-78, ISBN 978-1-4244-2819-9, San Jose, CA, USA, November 2008, IEEE/ACM.



Advances in Analog Circuits
Edited by Prof. Esteban Tlelo-Cuautle

ISBN 978-953-307-323-1
Hard cover, 368 pages
Publisher InTech
Published online 02, February, 2011
Published in print edition February, 2011

This book highlights key design issues and challenges to guarantee the development of successful applications of analog circuits. Researchers around the world share acquired experience and insights to develop advances in analog circuit design, modeling and simulation. The key contributions of the sixteen chapters focus on recent advances in analog circuits to accomplish academic or industrial target specifications.