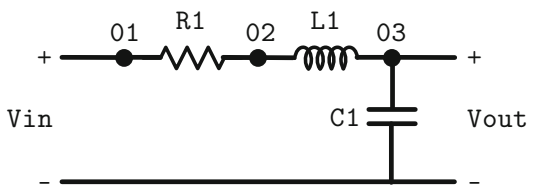
Chapter 5
Circuit analysis

High level design of VLSI systems assumes correct functionality of the underlying electrical circuits. Digital systems, for example, utilize clearly distinguishable binary signals. At lower abstraction layers, however, the electrical signals behave similar to analog signals. The electrical waveforms therefore satisfy stringent requirements, such as propagation delay, slew rate, and power dissipation, to satisfy signal integrity requirements. To evaluate these waveforms, circuit level analysis of VLSI systems is necessary. Due to the significant increase in the speed and complexity of integrated systems, accurate and computationally efficient circuit analysis has gained critical importance over the past decades.

Since the establishment of a mathematical structure for circuit theory in 1827 by G. S. Ohm [394], different methods for circuit analysis have been reported in the literature. A graph theoretic basis for circuit analysis was described in 1847 by G. R. Kirchhoff [256] by postulating two laws governing the current and voltage relationship within an arbitrary electrical circuit. By the late 19^{th} century, the theory of transient and alternating current was developed, incorporating the concepts of capacitance and inductance into the circuit analysis process [395]. Nonlinear circuit theory emerged in the early 20^{th} century, driven by the advent of nonlinear devices, particularly vacuum tubes [396].

Entering the era of integration, the need for accurate analysis of complex systems motivated the development of circuit simulation tools. Tensor analysis of electrical circuits, pioneered in 1934 by G. Kron [397], was a crucial precursor of early circuit simulators. The Transistor Analysis Program (TAP), developed in 1959 [398], is considered the earliest circuit simulation program [399]. Based on TAP, more advanced simulation tools were developed, including NET1 in 1963 [400] and SCEPTRE in 1967 [401], capable of handling a wide range of circuits, including both passive and nonlinear components. Important advancements in numerical integration, driven primarily by H. Shichman [402, 403], were vital in creating CIRcuit analysis PACkage (CIRPAC) [403] which exhibited an order of magnitude speedup as compared to other simulators of the time.

R. Bairamkulov, E. G. Friedman, *Graphs in VLSI*,
https://doi.org/10.1007/978-3-031-11047-4_5



a)

```
1. TEMP 300.0
2. GAIN 01 00 03 00 V/V AC
3. FREQ 10 0.1 100
4. R1 01 02 10.0
5. L1 02 03 200.0E-12
6. C1 03 00 10.0E-12
7. END
```

b)

Fig. 5.1 An example of AC analysis of a low pass filter using SLIC [261]. a) Target circuit, and b) SLIC code. Resistor `R1`, inductor `L1`, and capacitor `C1` are described on lines 4 to 6. Note the similarity with SPICE syntax. The `TEMP` statement specifies the operating temperature of the circuit. The `GAIN` statement specifies the output (`03` and `00`) and input (`01` and `00`) ports and the type of analysis (e.g., AC voltage transfer function). The range of frequencies is 0.1 to 100 MHz with ten points per decade, as specified in the `FREQ` statement on line 3. The input is terminated with the `END` statement.

The application of sparse matrix analysis to circuit simulation was a crucial advancement in the Advanced STatistical Analysis Program (ASTAP), developed in IBM in 1971 [404], significantly reducing memory requirements. Variable time step integration further improved accuracy and runtime by increasing or reducing the time resolution if the rate of change in the parameters is, respectively, high or low [404]. Other notable circuit simulators of the early 1970's include Computer Analysis of Nonlinear Circuits, Excluding Radiation (CANCER) [260], and Simulator for Linear Integrated Circuits (SLIC) [261], which utilized advanced linear algebraic methods for linearization, numerical stability, and accuracy control. An important feature of CANCER and SLIC was the user friendly input description language that contributed to the widespread adoption of these tools in both the industrial and academic communities. An example of a circuit described in the SLIC language is shown in Fig. 5.1. Note the similarity with current circuit simulation tools. Since an electrical circuit is fundamentally a graph, only connectivity information is required to describe a circuit, enabling efficient textual representation of the system.

The popularity of CANCER in the academic community motivated the development in 1973 of the open source Simulation Program with Integrated Circuit Emphasis, commonly known today as SPICE [399]. The second version of SPICE, released in 1975 [50], became the worldwide standard for circuit simulation. The success of SPICE2 can be largely attributed to the applicability of the tool to a wide range of linear and nonlinear circuits. This crucial feature of SPICE2 is achieved by utilizing modified nodal analysis (MNA), a robust method for numerical circuit analysis [405].

5.1 Modified nodal analysis

First presented in 1975 [64], MNA is a versatile method for analyzing linear circuits. The impedances, current sources, voltage sources, and nonlinear devices within a circuit are described in matrix form. If the conductance of each wire, voltage of each voltage source, and current of each current source is known, the potential difference across each edge (i.e., a circuit element, such as a resistor or current source) can be determined.

Suppose a circuit is represented by a directed multigraph $G = (V, E)$, the direction of the edges is arbitrary chosen, and the edge set is composed of five subsets,

$$E = E_v \cup E_i \cup E_r \cup E_c \cup E_l, \tag{5.1}$$

each representing, respectively, independent voltage sources, independent current sources, resistors, capacitors, and inductors. Recall from Subsection 3.4.1 that Y_d is the incidence matrix of a directed graph where an entry is

$$y_{n,e} = \begin{cases} 1, \text{ if the positive terminal of element } e \text{ connected to node } n & \text{(5.2a)} \\ -1, \text{ if the negative terminal of element } e \text{ connected to node } n & \text{(5.2b)} \\ 0, \text{ otherwise.} & \text{(5.2c)} \end{cases}$$

The elements within the network can be ordered such that

$$Y_d = \begin{bmatrix} Y_v & Y_i & Y_r & Y_c & Y_l \end{bmatrix}, \tag{5.3}$$

$$\mathbf{v} = \begin{bmatrix} \mathbf{v}_v \\ \mathbf{v}_i \\ \mathbf{v}_r \\ \mathbf{v}_c \\ \mathbf{v}_l \end{bmatrix}, \tag{5.4}$$

$$\mathbf{i} = \begin{bmatrix} \mathbf{i}_v \\ \mathbf{i}_i \\ \mathbf{i}_r \\ \mathbf{i}_c \\ \mathbf{i}_l \end{bmatrix}, \tag{5.5}$$

where $\mathbf{v} \in \mathbb{R}^{|E|}$ and $\mathbf{i} \in \mathbb{R}^{|E|}$ are vectors of, respectively, the voltage across and current through the corresponding element, and subscripts v, i, r, c, and l indicate the type of circuit element, respectively, the independent voltage and current sources, resistors, capacitors, and inductors. The elements of $\mathbf{i}_i \in \mathbb{R}^{|E_i|}$

represent the current through the independent current sources and are known *a priori*. The remaining current and voltage vectors are related via the following relationships [406],

$$\mathbf{i}_r = \mathcal{G}\mathbf{v}_r, \tag{5.6}$$

$$\mathbf{i}_c = \mathcal{C}\frac{d}{dt}\mathbf{v}_c, \tag{5.7}$$

$$\mathbf{v}_l = \mathcal{L}\frac{d}{dt}\mathbf{i}_l, \tag{5.8}$$

where $\mathcal{G} \in \mathbb{R}^{|E_r|\times|E_r|}$ and $\mathcal{C} \in \mathbb{R}^{|E_c|\times|E_c|}$ are diagonal matrices representing, respectively, the conductance and capacitance of the respective elements, and $\mathcal{L} \in \mathbb{R}^{|E_l|\times|E_l|}$ is the inductance matrix representing the self- and mutual inductance within a circuit. Note that $\mathcal{L}$ is a diagonal matrix if the mutual inductances are ignored.

The primary equation governing the static analysis of circuits without dependent sources can be formulated as

$$\begin{bmatrix} G & Y_v \\ Y_v^T & 0 \end{bmatrix} \mathbf{e} = Y_i \mathbf{i}_i, \tag{5.9}$$

where $\mathbf{e} \in \mathbb{R}^{|V|}$ is the vector of voltage at each node, and $G = Y_g \mathcal{G} Y_g^T$ is the conductance matrix of a resistive network.

By constructing and solving (5.9), the steady state voltage at each node can be determined. Practical VLSI circuits however contain circuit elements that display transient behavior. These elements include linear primitives, such as capacitors and inductors, and nonlinear elements, such as transistors and memristors. To model the behavior of these elements, numerical differentiation is applied. Each transient element is replaced by an equivalent circuit element called a *companion model* that includes resistors and independent sources. For example, the transient current $i_C(t)$ through a capacitor as a function of time t is

$$i_C(t) = C\frac{dv_C(t)}{dt}, \tag{5.10}$$

where C is the capacitance and v_C is the voltage across the capacitor. Discretization by the Backward Euler method yields

$$i_C(t^k) = \frac{C}{h}v_C(t^k) - \frac{C}{h}v_C(t^{k-1}), \tag{5.11}$$

where t^{k-1} and t^k are consecutive discrete time instants, and h is the time step. This expression is equivalent to

$$i_C(t^k) = g_{eq} v_C(t^k) + i_{eq}, \tag{5.12}$$

where g_{eq} is the equivalent instantaneous conductance of the capacitor,

$$g_{eq} = \frac{C}{h}, \tag{5.13}$$

and i_{eq} is the equivalent current source across the capacitor,

$$i_{eq} = -\frac{C}{h} v_C(t^{k-1}). \tag{5.14}$$

During transient analysis, a capacitor is replaced by an equivalent companion model, as shown in Fig. 5.2a. During each time step, the transient parameters within the model are adjusted, modeling the instantaneous behavior of the element.

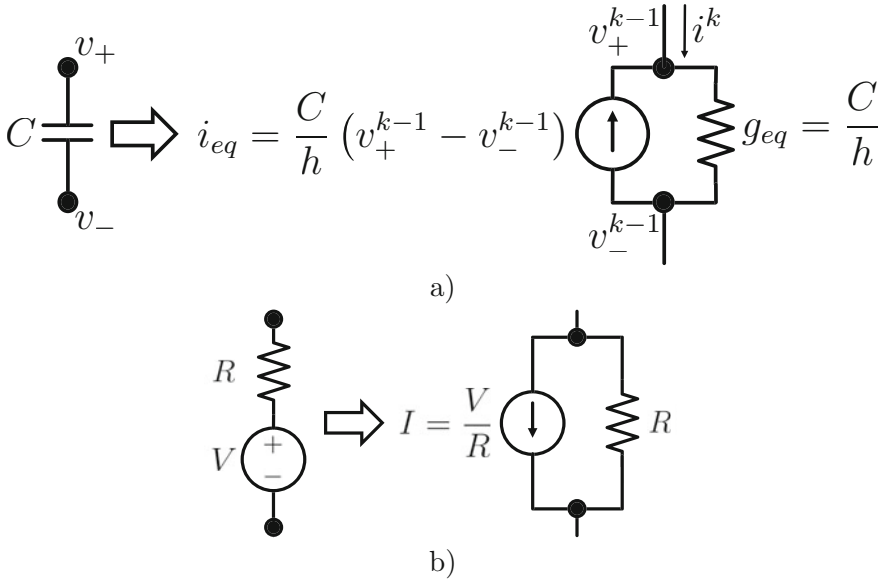


Fig. 5.2 Companion models for transient analysis of circuits. a) Capacitor model, and b) independent voltage source model

In matrix form, the companion models transform (5.9) into

$$\begin{bmatrix} A & Y_v \\ Y_v^T & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} C & 0 & 0 \\ 0 & L & 0 \\ 0 & 0 & 0 \end{bmatrix} \frac{d}{dt}\mathbf{x} = \begin{bmatrix} Y_i \mathbf{i}_i \\ 0 \end{bmatrix}, \tag{5.15}$$

or, in a more compact form,

$$\tilde{G}\mathbf{x} + \tilde{C}\dot{\mathbf{x}} = \mathbf{b} \tag{5.16}$$

where

$$A = \begin{bmatrix} G & Y_l \\ -Y_l^T & 0 \end{bmatrix}, \tag{5.17}$$

$$\mathbf{x} = \begin{bmatrix} \mathbf{e} \\ \mathbf{i}_l \\ \mathbf{i}_v \end{bmatrix}, \tag{5.18}$$

and $C = Y_c \mathcal{C} Y_c^T$ is the capacitance matrix [406].

Discretizing (5.15) yields

$$\left(\tilde{G} + \frac{2}{h}\tilde{C}\right)\mathbf{x}^k = \mathbf{b}^k + \mathbf{b}^{k-1} - \mathbf{x}^{k-1}\left(\tilde{G} - \frac{2}{h}\tilde{C}\right), \tag{5.19}$$

where k is the iteration number. Equation (5.19) is a system of linear equations. MNA-based transient analysis therefore requires an iterative solution of a system of linear matrix equations for each time step.

The primary advantage of MNA is versatility. Any linear circuit can be analyzed with MNA. To analyze nonlinear devices within a circuit, such as transistors, memristors, and magnetic tunnel junctions, linearized models are used [407, 408]. These models approximate the device behavior around a specific operating point. The computational and memory complexity of MNA is however of great concern. The runtime to solve a linear equation grows superlinearly with the number of nodes, requiring significant computational time for large systems, as in VLSI systems [145]. Furthermore, matrices $\tilde{G}$ and $\tilde{C}$ lose the symmetric positive definite (SPD) property in the presence of independent voltage sources. Efficient algorithms suited for SPD matrices, such as Cholesky factorization [409] or conjugate gradient method [410], can therefore no longer be used to solve (5.15), requiring more expensive algorithms such as LU factorization [411].

To preserve the SPD property, those circuit elements producing the voltage source and additional nodes within the network can be transformed using a Norton equivalent circuit to eliminate the voltage source and any associated rows and columns. An example of a Norton equivalent of an independent voltage source

connected in series with a resistor is shown in Fig. 5.2b [412]. By eliminating the independent voltage sources, (5.15) becomes

$$\begin{bmatrix} G & Y_l \\ -Y_l^T & 0 \end{bmatrix} \mathbf{e} + \begin{bmatrix} C & 0 \\ 0 & L \end{bmatrix} \frac{d}{dt}\mathbf{e} = Y_i \mathbf{i}_i, \tag{5.20}$$

yielding the SPD matrix $\left(\tilde{G} + \frac{2}{h}\tilde{C}\right)$ in (5.19) [406].

Despite restoring the SPD property, the analysis of large circuits requires significant computational resources. Direct linear matrix solvers place excessive demand on the memory during computations of extremely large networks. Alternative methods have been introduced to circumvent the superlinear complexity of the circuit analysis process. For example, a significant structural similarity exists between linear electrical circuits and finite element discretization of partial differential equations (PDE). Methods for accelerated solution of PDEs are therefore often applicable to the analysis of linear circuits. Many approaches utilize graph theory to accelerate the analysis process. A variety of techniques for fast circuit analysis is described in the upcoming sections.

5.2 Iterative numerical methods

Both DC and transient forms of MNA can be represented as a standard system of linear equations [65],

$$A\mathbf{x} = \mathbf{b}, \tag{5.21}$$

where $\mathbf{x}$ is the vector representing the voltage at each node and the current through each voltage source within a network, and $\mathbf{b}$ is the vector of the current being injected and the voltage sources. Network models of modern ICs are prohibitively large, disallowing the use of direct solution methods such as LU factorization or Cholesky factorization [411]. Iterative solvers, such as the conjugate gradient (CG) method [410] or generalized minimal residual method (GMRES) [413], should therefore be used to circumvent this limitation.

Reformulating (5.21) yields

$$\mathbf{b} - A\mathbf{x} = 0. \tag{5.22}$$

If vector $\mathbf{x}$ is replaced by vector $\mathbf{x}'$, (5.21) becomes

$$\mathbf{b} - A\mathbf{x}' = \mathbf{r}, \tag{5.23}$$

where $\mathbf{r}$ is called a residual. Observe that the norm of the residual $||\mathbf{r}||$ becomes smaller if $\mathbf{x}'$ is close to $\mathbf{x}$. Iterative linear equation solvers attempt to minimize $||\mathbf{r}||$ by iteratively adjusting vector $\mathbf{x}'$, thereby closely approximating the exact solution $\mathbf{x}$.

Classic iterative algorithms are *stationary* methods [414] that represent a system of equations as

$$\mathbf{x}^k = B\mathbf{x}^{k-1} - \mathbf{c}, \tag{5.24}$$

where $\mathbf{x}^k$ is the approximation of the solution after the k^{th} iteration, and matrix B and vector $\mathbf{c}$ depend upon the chosen iterative method. Note that matrix B and vector $\mathbf{c}$ remain invariant during the iterative process, hence the methods are called stationary [415]. Classical stationary methods operate by *splitting* matrix A into two matrices [415],

$$A = M - N. \tag{5.25}$$

The k^{th} iteration of these methods is

$$\mathbf{x}^k = M^{-1}\left(N\mathbf{x}^{k-1} - b\right). \tag{5.26}$$

Common iterative methods suggest different methods for splitting matrix A [416],

$$M = \begin{cases} D, & \text{Jacobi method} & \text{(5.27a)} \\ D - E, & \text{Forward Gauss-Seidel method} & \text{(5.27b)} \\ D - F, & \text{Backward Gauss-Seidel method} & \text{(5.27c)} \\ \dfrac{1}{\omega}D - E, & \text{Successive OverRelaxation (SOR) method,} & \text{(5.27d)} \end{cases}$$

where ω is the relaxation parameter, E and F are, respectively, the strictly lower and strictly upper triangular parts of A, and D is the diagonal part of A.

Advanced iterative methods are typically not stationary, i.e., the terms of (5.24) are not maintained constant. The CG method [410] is one of the most common non-stationary iterative method for solving systems of the form described by (5.21) when matrix A is symmetric positive definite (SPD). The algorithm is based upon the observation that the exact solution $\mathbf{x}$ minimizes a convex function,

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{b}^T\mathbf{x}. \tag{5.28}$$

The solution of (5.21) is determined by minimizing $f(\mathbf{x})$ via gradient descent [417]. The gradient of function $f(\mathbf{x})$ is

$$\nabla f(\mathbf{x}) = A\mathbf{x} - \mathbf{b}. \tag{5.29}$$

The solution is found by iteratively shifting $f(\mathbf{x})$ in the direction of steepest descent,

$$\mathbf{x}^{k+1} = \mathbf{x}^{k-1} - \alpha^k \left(A\mathbf{x}^{k-1} - \mathbf{b} \right), \tag{5.30}$$

or, alternatively,

$$\mathbf{x}^k = \mathbf{x}^{k-1} + \alpha^k \mathbf{r}^{k-1}, \tag{5.31}$$

where α^k is the step size during iteration k.

A limitation of the CG method is the limited applicability of the algorithm, since only SPD matrices are considered. Furthermore, the upper bound on the number of CG iterations before converging is equal to the size of the matrix, impractical for large systems. A large variety of solvers is proposed that partially or fully overcome these limitations, including the BIConjugate Gradient (BICG), BIConjugate Gradient STABilized (BICGSTAB), MINimal RESidual (MINRES), and Generalized Minimal RESidual methods (GMRES) [418].

Preconditioning is often used to accelerate the convergence of the iterative methods. During preconditioning, a linear matrix equation is transformed into

$$\tilde{A}\tilde{\mathbf{x}} = \tilde{\mathbf{b}} \tag{5.32}$$

where

$$\tilde{A} = M_1^{-1} A M_2^{-1}, \tag{5.33}$$

$$\tilde{\mathbf{x}} = M_2 \mathbf{x}, \tag{5.34}$$

$$\tilde{\mathbf{b}} = M_1^{-1} \mathbf{b}, \tag{5.35}$$

and $M = M_1 M_2$ is a nonsingular matrix called a *preconditioner* [418]. After solving (5.32), (5.34) is solved to determine $\mathbf{x}$. Proper choice of the preconditioner accelerates the convergence of the iterative solvers, ensuring that determining M and solving (5.32) and (5.34) are more efficient than solving the original system. The computational cost of producing the preconditioner should however be small since the difficulty in producing matrix M negates the computational benefits.

Matrix M, described in (5.25), and (5.27a) to (5.27d) can be used as a preconditioner during the circuit analysis process [416]. Several features make methods (5.27a) to (5.27d) attractive. Systems involving matrix M are relatively easy to solve, since M is either diagonal or triangular. Systems produced in practical VLSI systems are sparse diagonally dominant matrices. The diagonal elements of a Laplacian matrix of an underlying graph are typically larger than the non-diagonal elements. The number of nodes of an underlying circuit graph is proportional to the

number of edges, since most nodes are only connected to the immediate neighbors. Systems composed of sparse diagonally dominant matrices are well suited for preconditioning using split matrices, significantly accelerating the convergence process [418].

Other popular preconditioning approaches exist that include incomplete factorization and approximate inverse. Incomplete LU (ILU) factorization, for example, is based on approximating $A \approx \tilde{L}\tilde{U}$ [416], where $\tilde{L}$ and $\tilde{U}$ are sparse upper and lower triangular matrices, yielding a preconditioned system,

$$\tilde{L}^{-1}A\tilde{U}^{-1}(\tilde{U}\mathbf{x}) = \tilde{L}^{-1}\mathbf{b}. \tag{5.36}$$

Incomplete Cholesky decomposition is a similar procedure restricted to positive definite matrices where the sparse approximation $A \approx R^T R$ is determined. The SParse Approximate Inverse (SPAI) [419] preconditioner exhibits performance superior to incomplete factorization methods [420] when applied to diagonally dominant problems.

The iterative methods and preconditioners described in this section are considered general purpose, effectively handling a wide range of problems while significantly reducing the memory requirements. Superior performance can however be achieved by applying advanced analysis methods, exploiting special features of practical circuit graphs, such as sparsity, smoothness, and graph partitioning. The upcoming subsections describe enhancements to MNA-based circuit analysis, including domain decomposition, multigrids, and hierarchical matrices.

5.2.1 Domain decomposition

Due to the superlinear complexity of linear system solvers, the divide-and-conquer approach [421] can be effective in tackling these problems. Two advantages make divide-and-conquer algorithms particularly attractive for circuit analysis. If solving a problem requires $O(n^p)$ time, where n is the problem size and $p > 1$, decomposing the problem into m sequentially solved parts yields a runtime of $O\left(\frac{n^p}{m^{p-1}}\right)$, assuming negligible computational overhead. Due to the superlinear complexity of linear system solvers, i.e., $p > 1$, this approach can be effective in reducing the computational burden. Furthermore, these m parts can be processed in parallel, further reducing the runtime.

Domain decomposition (DD) is one of the most successful divide-and-conquer strategies for circuit analysis. The main principle of the DD technique is partitioning a circuit graph $G = (V, E)$ into multiple subgraphs $G_i = (V_i, E_i), i \in \{1, \ldots, m\}$ and a subgraph of interface nodes $G_0 = (V_0, E_0)$. An illustrative example is shown in Fig. 5.3. Observe that the interface nodes do not belong to any subgraph, and no node belongs to more than one partition. Note that this limitation not only prohibits conduction between the subgraphs, but also forbids capacitive and

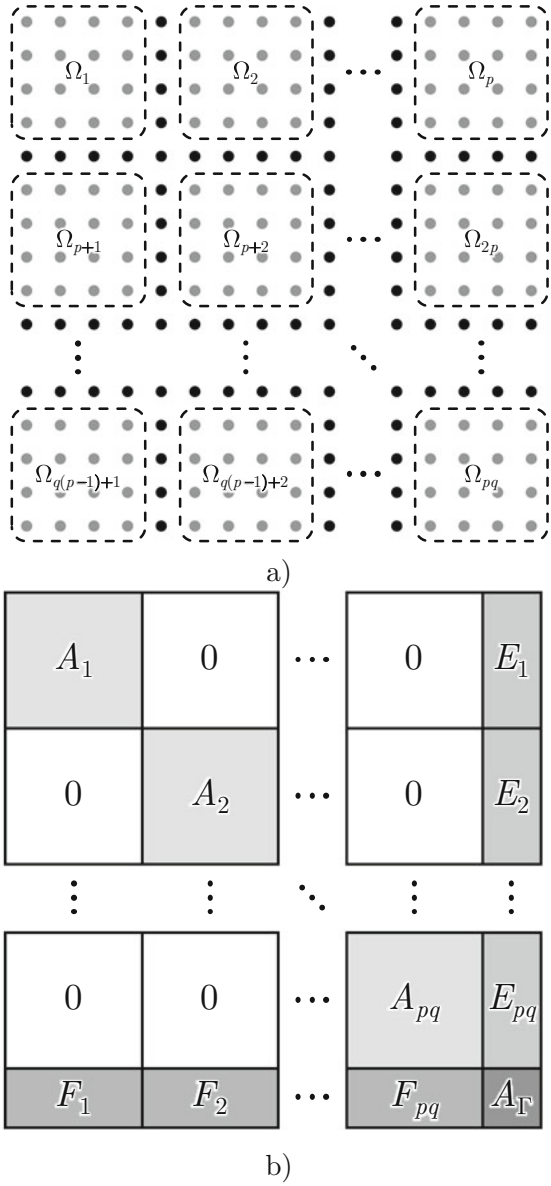


Fig. 5.3 Domain decomposition process within a grid. a) A large mesh is divided into $m = pq$ subdomains $\{\Omega_1, \ldots, \Omega_{pq}\}$ (gray nodes) and interfaces (black nodes). b) Connectivity within domain i is described by matrix A_i, while connections with the interface are described in E_i and F_i. A_Γ encodes the connectivity within the interface. The dimensions of A_Γ are typically smaller than the dimensions of A_i.

inductive coupling between subdomains. The standard linear equation of (5.21) can therefore be represented as an "arrowhead matrix" [422],

$$\begin{bmatrix} A_1 & 0 & \dots & 0 & E_1 \\ 0 & A_2 & \dots & 0 & E_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & A_m & E_m \\ F_1 & F_2 & \dots & F_m & A_0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_m \\ \mathbf{x}_0 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_m \\ \mathbf{b}_0 \end{bmatrix}, \tag{5.37}$$

where A_i represents the connectivity within subgraph G_i, E_i and F_i represent the connectivity between G_i and interface G_0, A_0 represents the connectivity among the interface nodes, $\mathbf{x}_i$ denotes the unknown voltages and currents within G_i, and $\mathbf{b}_i$ is the vector of current and voltage sources connected to G_i. Equation (5.37) can be split into two parts,

$$A\mathbf{x} + E\mathbf{x}_0 = \mathbf{b}, \tag{5.38}$$

$$F\mathbf{x} + A_0\mathbf{x}_0 = \mathbf{b}_0, \tag{5.39}$$

where A is a block diagonal matrix produced from subgraph matrices A_i, and F, E, $\mathbf{x}$, and $\mathbf{b}$ are produced by concatenating F_i, E_i, $\mathbf{x}_i$, and $\mathbf{b}_i$. Solving (5.38) for $\mathbf{x}$ and substituting into (5.39) yields

$$\mathbf{x} = A^{-1}(\mathbf{b} - E\mathbf{x}_0), \tag{5.40}$$

$$(A_0 - FA^{-1}E)\mathbf{x}_0 = \mathbf{b}_0 - FA^{-1}\mathbf{b}. \tag{5.41}$$

To solve (5.41), $P = A^{-1}E$ and $\mathbf{q} = A^{-1}\mathbf{b}$ are determined. Due to the block diagonal structure of A, these equations can be decomposed into m independent equations,

$$P_i = A_i^{-1}E_i, \tag{5.42}$$

and

$$\mathbf{q}_i = A_i^{-1}\mathbf{b}_i. \tag{5.43}$$

Matrix P and vector q are substituted into (5.41), yielding

$$(A_0 - FP)\mathbf{x}_0 = \mathbf{b}_0 - F\mathbf{q}. \tag{5.44}$$

Since the size of the interface set $|V_0|$ is typically much smaller than the size of any subgraph $|V_i|$, (5.44) requires relatively small computational resources. Equation (5.40) is transformed into

$$\mathbf{x} = \mathbf{q} - P\mathbf{x}_0. \tag{5.45}$$

Similar to (5.42) and (5.43), (5.45) can be decomposed into m independent systems,

$$\mathbf{x}_i = \mathbf{q}_i - P_i\mathbf{x}_0. \tag{5.46}$$

Due to the mutual independence of these expressions, (5.42), (5.43), and (5.46) can be solved in parallel. Furthermore, due to the small dimensions (i.e., relatively few interface nodes), the total runtime to solve m systems is smaller than the runtime to solve the original system, yielding additional performance improvement.

One of the earliest applications of domain decomposition in circuit analysis is discussed in [423]. A DRAM system composed of 130,000 transistors was successfully analyzed using 27 connected workstations operating in parallel. On-chip power delivery system analysis using domain decomposition is proposed in [422]. Domain decomposition combined with direct LU factorization of the subdomains achieved the maximum performance in case studies, completing a DC analysis of a ten million node system in 450 seconds.

The major advantage of domain decomposition is parallelization. Increasing the number of subdomains however increases the size of the interface graph G_0, potentially negating any performance gains. Overlapping domain decomposition modifies the original non-overlapping technique by allowing partitions to overlap [424], as illustrated in Fig. 5.4. The combined analysis of overlapping domains is based on the Schwarz method [425] and is used in [426] to complete the analysis of a power grid with 192 million nodes in five minutes while utilizing 1,200 processors operating in parallel.

5.2.2 $\mathcal{H}$-matrix

Another divide-and-conquer approach to circuit analysis is the application of the hierarchical matrix ($\mathcal{H}$-matrix) technique. Assume the target graph $G = (V, E)$ is described by a nodal analysis matrix A. The method commences with hierarchical clustering of the entries within matrix A, yielding cluster tree T, as illustrated in Fig. 5.5d. The first step of top-down clustering [427] splits matrix A into multiple submatrices,

$$A = \begin{bmatrix} A_{1,1} & \dots & A_{1,m} \\ \vdots & \ddots & \vdots \\ A_{m,1} & \dots & A_{m,m} \end{bmatrix}. \tag{5.47}$$

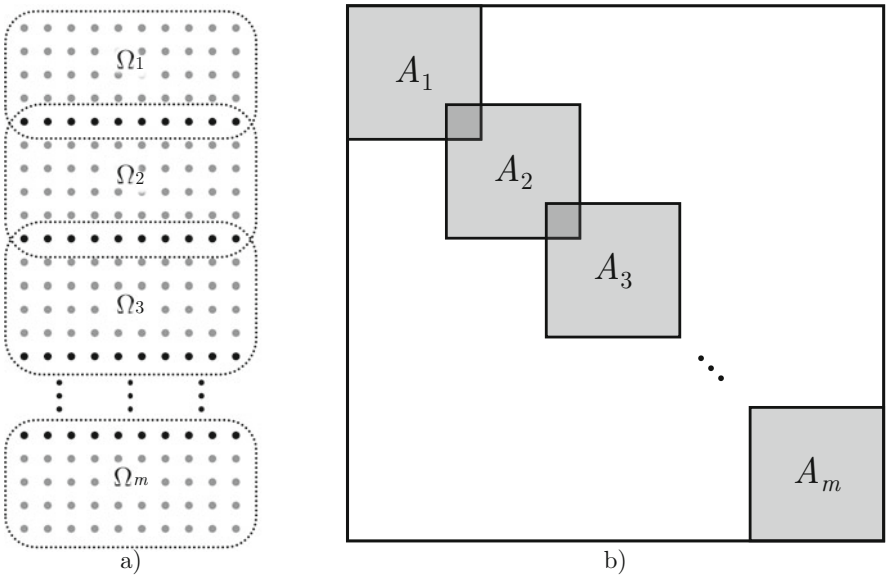


Fig. 5.4 Overlapping domain decomposition technique. a) Overlapping domains within the mesh. The black dots represent shared nodes. b) Resulting MNA matrix with overlapping sections corresponding to shared nodes.

Blocks $A_{i,j} \in \mathbb{R}^{p \times q}$ become the children of the root node of cluster tree T. The diagonal blocks $A_{i,i}$, $i \in \{1, \ldots, m\}$ are typically full rank matrices, while the off diagonal blocks are rank deficient. If rank $k_{i,j}$ of block $A_{i,j}$ is smaller than the specified threshold $k_{\min}$, the matrix can be efficiently factorized as the product of two small matrices,

$$A_{i,j} = MN^T, \tag{5.48}$$

where $M \in \mathbb{R}^{p \times k}$, $N \in \mathbb{R}^{q \times k}$, and $k \ll p, q$. Any block within T is split if the size of the block is greater than the specified minimum size $m_{\min}$ and if the rank of the matrix is greater than $k_{\min}$. Otherwise, the block is not split and is stored in factored form.

The main purpose of an $\mathcal{H}$-matrix is a cluster tree representation of matrix A. The resulting block matrix is illustrated in Fig. 5.5c. Those leaves stored in factored form require relatively low processing runtime. These features enable an efficient approximation of the LU factorization and inverse of matrix A, yielding significant improvement in runtime and memory. For example, the complexity of LU factorization is reduced from $O\left(n^3\right)$ to $O\left(n\left(\log n\right)^2\right)$. Partial element equivalent circuit analysis of a power supply layout is presented in [428], achieving up to four orders of magnitude speedup and up to a 50 fold reduction in memory requirements. The compatibility of the $\mathcal{H}$-matrix with finite element analysis

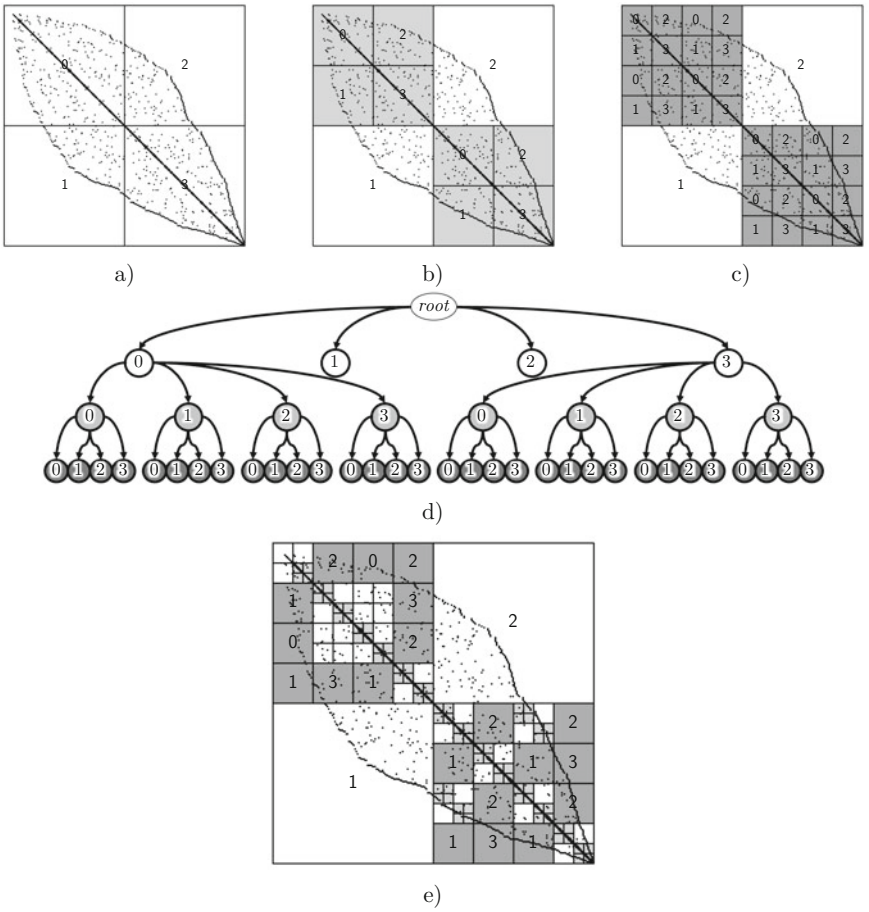


Fig. 5.5 Construction of an $\mathcal{H}$-matrix of a sparse matrix describing a circuit with 256 nodes and 471 edges. The black dots represent the sparsity pattern of the original matrix. a) The original matrix is divided into four submatrices. b) Submatrices 1 and 2 are significantly rank deficient and are therefore not divided. Submatrices 0 and 3 are further split into four submatrices. c) $\mathcal{H}$-matrix after the third iteration. d) Cluster tree T after three iterations. e) Final $\mathcal{H}$-matrix after five iterations. The densest regions along the diagonal are split until the minimum submatrix size is achieved.

enables efficient fine grained thermal analysis of a three-dimensional IC with more than a million discrete points [429].

5.2.3 Multigrid methods

The earliest works on multigrids date back to the 1960's when R. P. Fedorenko suggested doubling the mesh spacing to solve the Poisson's equation [430]. This approach allowed an approximate solution of a coarse grid to be efficiently determined. The coarse grid solution is subsequently mapped onto the original grid, providing a good initial point for solving the original grid. An order of magnitude reduction in the number of iterations was reported when using a coarser grid [430]. The method, applied in [430], was subsequently formalized in the 1970's by A. Brandt [431] and W. Hackbusch [432].

Three cornerstone operations constitute the multigrid method, namely *smoothing*, *restriction*, and *prolongation*. Fundamentally, smoothing is the partial application of an iterative solver, as described in Section 5.2. Several iterations of an iterative solver significantly reduce the residual error, thereby shifting the approximate solution closer to the exact solution. Consider, for example, a one-dimensional Poisson's equation,

$$f''(x) = \sin(x), \tag{5.49}$$

with boundary conditions $f(0) = f(1) = 0$. The problem is discretized using the trapezoidal rule,

$$\frac{f(x+h) - 2f(x) + f(x-h)}{h^2} = \sin(x), \tag{5.50}$$

where h is the discretization step. In matrix form, the equation becomes

$$\begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x} = \begin{bmatrix} \sin(0) \\ \sin(h) \\ \sin(2h) \\ \vdots \\ \sin(1-h) \\ \sin(1) \end{bmatrix}. \tag{5.51}$$

The system is solved using the Loose GMRES (LMGRES) method [433]. Convergence of the algorithm is depicted in Fig. 5.6. Observe that any difference between the adjacent points is quickly reduced, i.e., the high frequency errors are significantly dampened during each iteration.

Eliminating low frequency errors however requires significantly longer runtime. To overcome this issue, a restriction step is performed to coarsen the domain of the system. The frequency of the error is therefore effectively increased, permitting efficient elimination by smoothing. Formally, restriction of function $f : E \to F$ is function $f|_A : A \to F$, where $A \subset E$ and $f(x) = f|_A(x) \forall x \in A$. In the context of multigrids, this operation is effectively coarsening, reducing the number of points

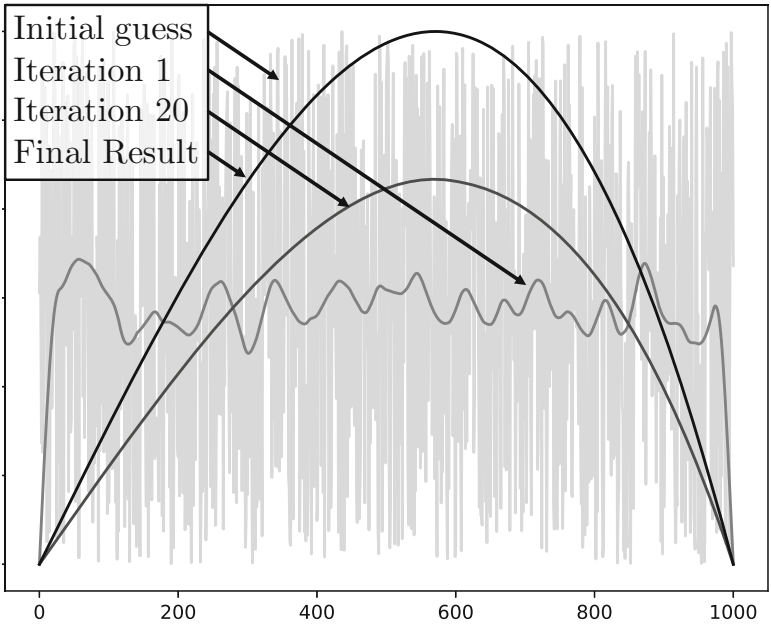


Fig. 5.6 Convergence of Poisson's equation using LMGRES method. The one-dimensional space is discretized using 1,001 points. The initial guess is a vector of random numbers. After the first iteration, the high frequency components of the initial vector are significantly reduced, producing a smoother curve. To eliminate the remaining low frequency components, additional iterations are necessary. In this example, the solution is achieved after 62 iterations.

within the grid. Suppose the grid is described as a graph $G_0 = (V_0, E_0)$, and function $v : V \to \mathbb{R}$ maps each node to the node voltage. The goal of the coarsening operation is a reduced version of an initial grid $G_1 = (V_1 \subset V_0, E_1)$, where the voltage within the original system $v_0(n)$ is equal to the voltage within coarse system $v_1(n)$ at any node $n \in V_1$. To recover the original solution from the approximate coarse solution, a prolongation operation is performed. Using interpolation, the solution of a coarse grid is mapped onto a fine grid. The resulting vector is typically a close approximation of a solution requiring few iterations to converge.

A single restriction procedure is often insufficient for significant acceleration. The restriction process is therefore repeatedly applied to further reduce the size of the mesh. Multiple prolongation operations are therefore necessary to recover the original grid. These procedures are formalized in *cycles* where a system undergoes a series of restrictions, prolongations, and smoothing. The most common cycles are V-cycle, W-cycle, and F-cycle [434], as illustrated in Fig. 5.7. These techniques tradeoff robustness with computational speed. The V-cycle is typically faster than the other cycle types but may fail to converge to a correct result. In contrast, the F-cycle requires more operations but is highly robust and accurate.

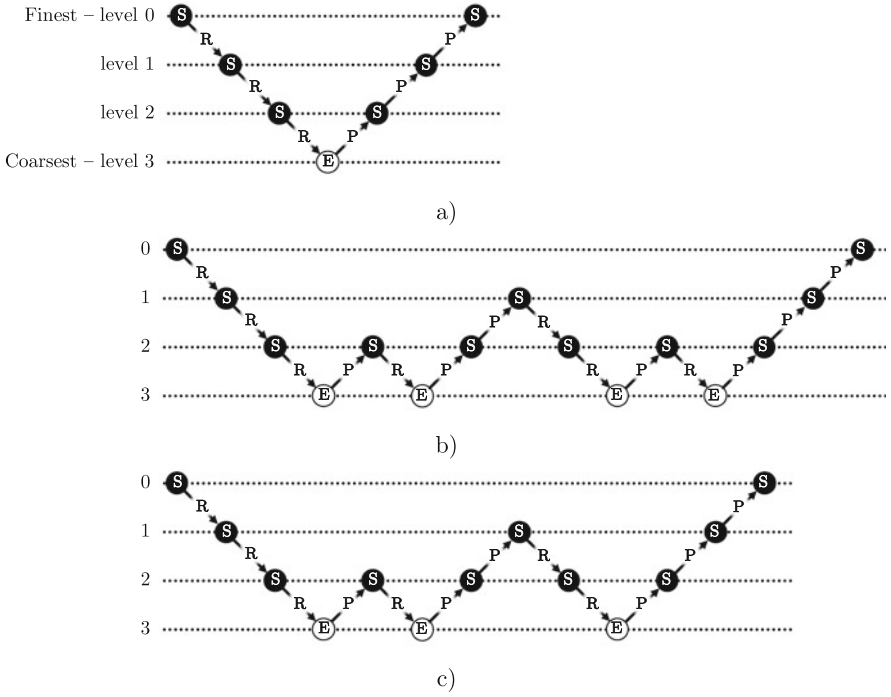


Fig. 5.7 Multigrid cycles. a) V-cycle, b) W-cycle, and c) F-cycle. 'S', 'R', and 'P' denote, respectively, the smoothing, restriction, and prolongation operations. At the coarsest level, a system is typically solved using an exact solver. This operation is denoted by 'E.'

One of the earliest applications of geometric multigrids to VLSI systems is described in [435]. A relatively straightforward application of geometric multigrids in [435] yields a 300 fold improvement in runtime with a peak error of over 20%. Geometric multigrids are highly suitable for analyzing regular physical layouts, supporting the efficient analysis of circuits with tens of millions of nodes [436].

The theoretical computational complexity of the geometric multigrid is $O(|V|)$ [437], an attractive feature in the analysis of large systems. A major limitation of geometric multigrids is reliance on the structural regularity of the problem domain. The algebraic multigrid (AMG) is an important generalization of geometric multigrids where no structural information is required for restriction and prolongation. One of the earliest applications of AMG to circuit analysis is proposed in [438], where a 16 to 20 fold improvement in runtime is achieved. Another notable result is PowerRush, an AMG-based DC and transient simulator exhibiting linear complexity [439, 440]. Up to nine levels of grid reduction are reported in [439], completing the analysis in 169 seconds after reducing a circuit with 38 million nodes to 264 nodes. The efficiency of AMG simulation enables large scale circuit optimization. For example, in [441], decoupling capacitor allocation is performed using AMG, optimizing circuits with up to a million nodes.

Restriction, smoothing, and prolongation are highly parallelizable due to the small number of steps with few dependencies [442]. These features enable GPU-based geometric multigrid acceleration of the circuit analysis process, achieving up to two orders of magnitude speedup [442, 443].

5.3 Non-MNA techniques

MNA and associated enhancements enable the efficient analysis of a wide range of complex circuits. Alternative techniques however exist that avoid MNA-based equations, often yielding superior performance as compared to MNA-based methods. Three techniques are presented in this section, namely, scattering parameters, random walks, and lattice graph analysis.

5.3.1 Scattering parameters

The detailed structure of an IC component is often unknown. This situation frequently occurs in two cases. If the components of the integrated system are supplied by a third party vendor, the internal structure of the components is treated as intellectual property (IP) and is typically not described or is purposely obfuscated [444, 445]. The structure of a component can also be highly complex, complicating the construction of a distributed model [75]. A scattering parameter (S parameter) model is often utilized in these cases, characterizing the frequency response of a circuit to input stimuli without revealing the internal structure (a black box). Examples of an S parameter model with two and n ports are depicted in Fig. 5.8. Parameters a_k and b_k correspond to normalized power waves [446],

$$a_k = \frac{1}{2g_k}\left(V_k + I_k Z_k\right), \tag{5.52}$$

$$b_k = \frac{1}{2g_k}\left(V_k - I_k Z_k^*\right), \tag{5.53}$$

where Z_k is the reference impedance at port k. Z_k^* denotes the complex conjugate of Z_k and

$$g_k = \sqrt{|\Re(Z_k)|}. \tag{5.54}$$

By measuring the response b_m of a circuit at port m in response to a unit power wave at port k, scattering parameter $s_{k,m}$ is

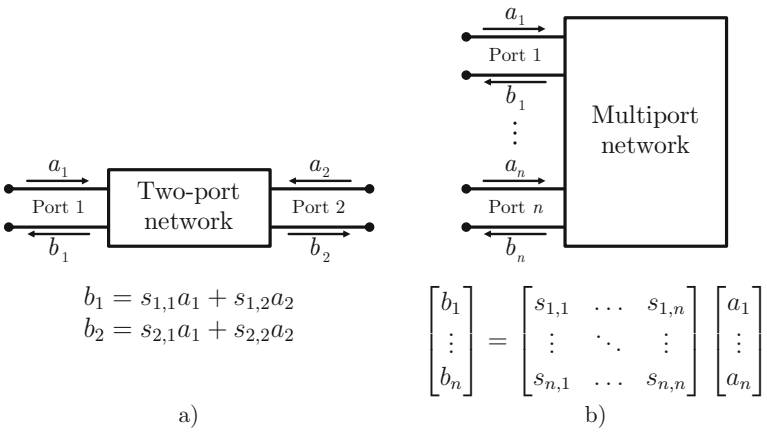


Fig. 5.8 Scattering parameter (S parameter) model of a component. a) Two port network, and b) multiport network. The component is treated as a black box with no knowledge of the internal structure, as opposed to a grey or white box utilizing, respectively, partial or complete structural information [447]. By applying stimuli at different ports of the components, the response of the system at each port is determined. The relationship between an excitation at port i and the response at port j is described by S parameters.

$$s_{k,m} = \frac{b_m}{a_k}. \tag{5.55}$$

In a multiport network, scattering parameter matrix S is produced that describes the relationship among the signals at different ports,

$$\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} s_{1,1} & s_{1,2} & \dots & s_{1,n} \\ s_{2,1} & s_{2,2} & \dots & s_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ s_{n,1} & s_{n,2} & \dots & s_{n,n} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}. \tag{5.56}$$

Note that any scattering parameter is a function of frequency. Measurements should therefore be performed at different frequencies to evaluate the response over the entire bandwidth of interest.

A major advantage of the S parameter model is the applicability of the model to an arbitrary system. The S parameter model requires no information describing the internal structure of the system. Furthermore, based on an S parameter matrix, other electromagnetic characteristics of a system can be determined [446]. Based on open circuit impedance (Z) parameters, for example, crucial parameters can be determined such as the self- and mutual inductances within a network [332].

$$Z = G_0^{-1} (I - S)^{-1} \left(S Z_0 - Z_0^* \right) G_0, \tag{5.57}$$

where I is the identity matrix,

$$G_0 = \begin{bmatrix} g_1 & 0 & \dots & 0 \\ 0 & g_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & g_n \end{bmatrix}, \tag{5.58}$$

and

$$Z_0 = \begin{bmatrix} Z_1 & 0 & \dots & 0 \\ 0 & Z_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & Z_n \end{bmatrix}. \tag{5.59}$$

Other parameters widely used during the design of analog circuits, such as Y, $ABCD$, or h parameters [446], can also be derived from the S parameters.

5.3.2 Random walks

A random walk is a stochastic process that describes a succession of steps of an object within a mathematical space [448]. A classic example of a random walk is a random walk along a one-dimensional integer axis, as illustrated in Fig. 5.9a. The particle is initially at position 0 and, every time step, the particle moves in a random direction. Different types of space and probability distributions of the transitions in a random walk exist, such as a discrete two-dimensional space, continuous two-dimensional space with a variable step length (such as Lévy Flight [449]), or a biased, continuous walk in three-dimensional space (see Figs. 5.9b to 5.9d). Common issues relating to a random walk include the expected distance of an object from the source after n steps, probability of a return to the origin after n steps, and probability of reaching a before reaching b, where a and b are arbitrary points within the space.

Manifestations of a random walk in physical systems have been studied before this term was first coined. In 1880, Lord Rayleigh studied the amplitude of oscillations due to multiple strings vibrating at the same frequency with a random phase [450]. This problem is analogous to a random walk on a one-dimensional axis. The erratic movement of dust particles, what will later be called Brownian motion, was discovered as early as 1784 by the Dutch scientist, J. Ingen-Housz [451]. A formal study of random walks has been applied to different physical phenomena, including diffusion in molecular physics [452], genetic drift in genetics [453], and measuring certain features of the World Wide Web [454, 455].

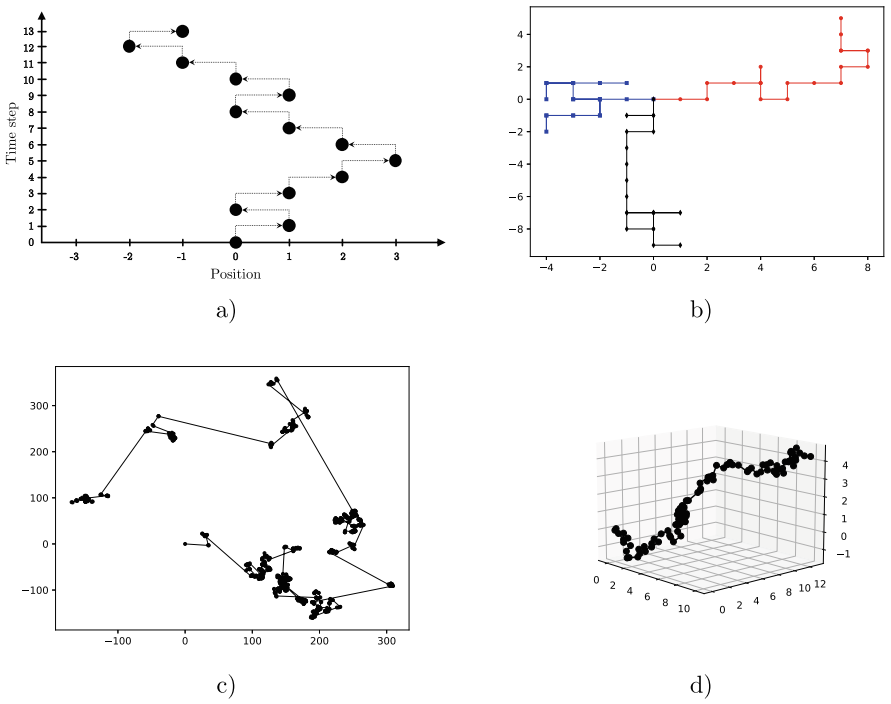


Fig. 5.9 Examples of a random walk. a) Discrete time one-dimensional random walk on an integer axis. b) Three unbiased random walks within a two-dimensional integer grid. c) Unbiased random walk within a continuous two-dimensional space. The direction of the step is uniformly random. The step size follows a Cauchy distribution. This type of random walk is commonly referred to as Lévy Flight [449]. d) Biased random walk within a three-dimensional integer space. The probability of a transition toward $+\infty$ is greater than the probability of a transition toward $-\infty$ along the x, y, and z axes.

One of the most extensively studied spaces of a random walk is a graph, where a particle moves towards the neighboring vertex at each time step. The probability of moving from vertex a toward vertex b is proportional to the weight of the edge (a, b). The analogy between a random walk and an electrical network was studied by C. St. J. A. Nash-Williams in [456]. The random walk equivalent of the effective resistance between a and b is the commute time between nodes a and b, *i.e.*, the expected number of steps in a random walk starting at a, visiting b, and returning to a (see Fig. 5.10). The conductance of a resistor is equivalent to the weight of an edge within a network. The probability of a transition along a specific edge is proportional to the weight of that edge. A particle moving in a random walk is therefore less likely to transition along a high resistance edge (or path).

The analogy between electrical circuits and random walks can be exploited in the analysis of electrical circuits. Different simulation tools based on random walks have been explored in the literature. By performing a random walk experiment

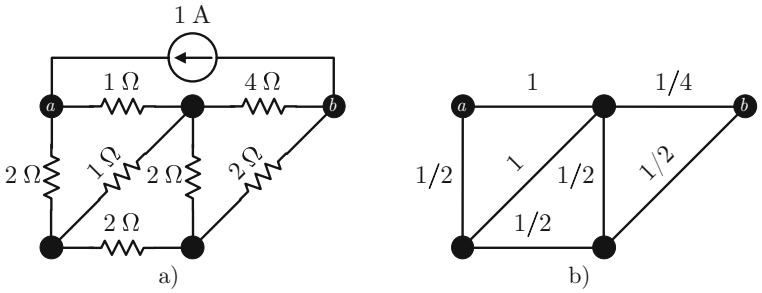


Fig. 5.10 Effective resistance between arbitrary nodes a and b is equivalent to the expected number of steps for a random walk to visit node b while starting and returning to node a, a) electrical circuit, and b) equivalent graph. The probability of transitioning towards a neighboring node is proportional to the conductance of the corresponding edge.

multiple times, the average number of steps converges towards the commute time which corresponds to the effective resistance. The earliest application of a random walk to linear circuit analysis is described in [70]. A major advantage highlighted in [70] is the linear relationship between circuit size and computational complexity. Different circuit simulation tools have been described in the literature, achieving a significant speedup as compared to conventional circuit analyses [70, 457–459]. Another aspect is the locality of the random walk. If the target nodes are located close to each other within a network, the random walk is more likely to terminate while exploring only a small portion of the network. This result is highly desirable when studying system perturbations, since only the affected portion of the system is analyzed. This advantage is exploited in [460] in the analysis of incremental changes in power grids.

A major issue pertaining to random walk-based simulation tools is the number of random walk experiments required to achieve a sufficiently small error. The error ε of a random walk is inversely proportional to the square root of the number of experiments M,

$$\varepsilon \propto \frac{1}{\sqrt{M}}. \tag{5.60}$$

To reduce the error by 50%, the number of experiments should be increased four fold. To overcome this issues, the 'importance sampling' technique is introduced in [461], significantly improving the speed of convergence. Another challenge of random walk-based tools is the possibility of excessively long walks, negating any computational speedup [70]. This issue can be eliminated by limiting the length of the random walk. The accuracy of the solution is however degraded by limiting the length.

Random walks are found in a wide variety of VLSI applications. A sensitivity analysis of VLSI power networks [462], for example, is a notable application, where the critical parameters affecting a power grid are evaluated. Matrix preconditioning based on random walks is described in [463], significantly accelerating the circuit

analysis process. Other notable applications of random walks in VLSI include modeling of thermal behavior [464], decoupling capacitor placement [465], and electromigration analysis [466].

5.3.3 Lattice graph

Due to the large scale of VLSI systems, the physical structures are often highly regular, composed of millions to billions of identical elements distributed within a system. The on-chip power grid is a prominent example of a regular structure composed of two or more layers of identical interconnects. An example of a power grid is shown in Fig. 5.11a. Grids are highly reliable due to the many redundant paths. The number of paths connecting two corners of a grid is [467]

$$\binom{x+y}{x} = \frac{(x+y)!}{x!y!}, \tag{5.61}$$

where x and y are, respectively, the horizontal and vertical dimensions of a grid. The number of paths in (5.61) grows superlinearly with x and y, yielding a high degree of redundancy even in relatively small grids. The failure of a single or multiple wire segments can be tolerated since the remaining wires provide the necessary connections. Additional benefits of a grid include shielding and decoupling that reduce parasitic capacitive and inductive coupling in global clock and data lines [468]. A grid structured power network can be modeled as a two-dimensional resistive lattice, as shown in Fig. 5.11b [469]. Depending upon the metal pitch and die size, the dimensions of a grid can vary from hundreds to tens of thousands of segments [72]. The large dimensions enable the use of infinite mesh methods for analyzing grid structured power networks.

Multidimensional mesh structures assuming infinite mesh dimensions have been extensively studied in the literature. In 1936, W. H. McCrea studied the following problem [470],

> *"In a rectangular lattice, at every time instant a point P moves from one lattice point to one of the neighboring points. Each adjacent point has equal probability of being selected. Determine the probability that the particular boundary point is ultimately reached."*

Variations of this problem on different two- and three-dimensional lattices were solved by W. H. McCrea and F. J. W. Whipple in 1940 [471], where different finite and infinite rectangular lattices are analyzed. A notable result is an expression for the average flow of particles between the source and target points within an infinite two-dimensional lattice,

$$G(x, y) = \frac{2}{\pi} \int_0^{\pi} \frac{1 - \cos(\lambda y) \exp(-\mu |x|)}{\sinh(\mu)} d\lambda, \tag{5.62}$$

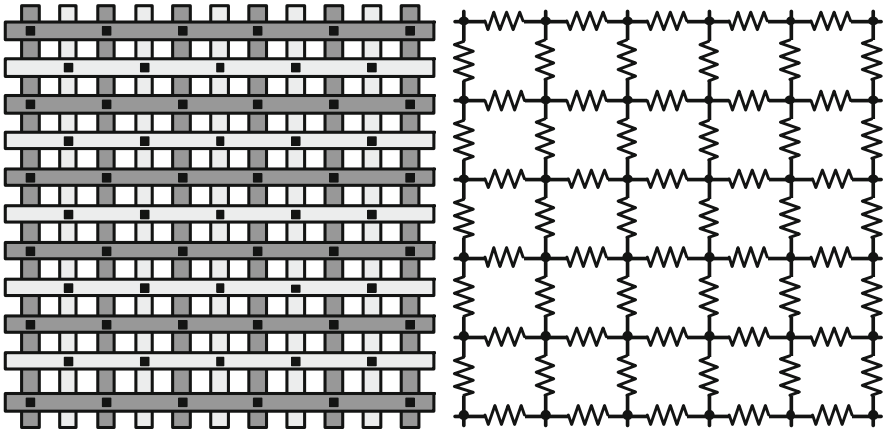


Fig. 5.11 On-chip power grid. a) Layout of power (dark grey) and ground (light grey) distribution networks, and b) power distribution network modeled as a resistive lattice.

where $\cos(\lambda) + \cosh(\mu) = 2$, and x and y denote the number of resistors separating the source and target points in, respectively, the horizontal and vertical direction.

The link between random walk and circuit theory was not widely recognized in the 1940's. An electrical formulation of the problem solved by W. H. McCrea and F. J. W. Whipple in [471] is

> *Determine the effective resistance between two arbitrary points* (x_0, y_0) *and* (x, y) *within a two dimensional grid of resistors with resistance* r *(see Fig. 5.11b)*

An easier problem of determining the effective resistance between *adjacent* nodes in an infinite resistive lattice was solved in 1949 [472] based on the principles of symmetry and superposition. Suppose current $4i$ is injected at an arbitrary node a, as illustrated in Fig. 5.12a. Due to symmetry, the current through each of the four adjacent branches is i. Now withdraw current $4i$ from neighboring node b. The current though each adjacent branch is also i, as shown in Fig. 5.12b. By superimposing these solutions, the current through a resistor connecting a and b is $2i$, as illustrated in Fig. 5.12c. The effective resistance is found by equating the voltage drop across resistor ab with the voltage drop across the effective resistance of the grid,

$$4i\,R_{eff} = 2ir, \tag{5.63}$$

yielding

$$R_{eff} = \frac{r}{2}. \tag{5.64}$$

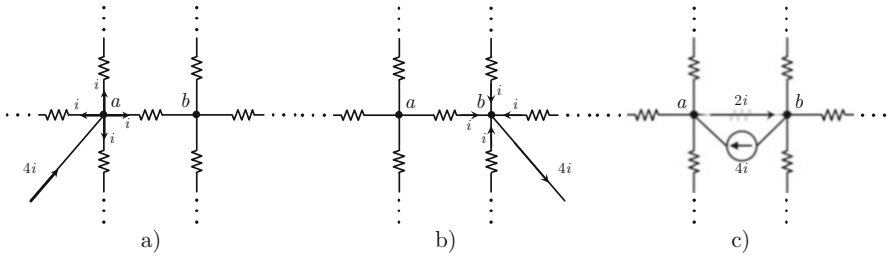


Fig. 5.12 Superposition applied to determine the effective resistance between adjacent nodes a and b in an infinite two-dimensional resistive lattice. a) Current $4i$ is injected into node a. Due to symmetry, the current through each adjacent resistor is i, flowing away from node a. b) Current $4i$ is drawn from node b. Due to symmetry, the current through each adjacent resistor is i, flowing towards node b. c) Superposition of current injection and withdrawal. The current through resistor ab is $2i$.

Despite the relative simplicity of the problem for adjacent nodes, the general problem requires advanced mathematical methods. Different alternative solutions to the problem of determining the effective resistance within a grid have been presented in the literature [472] in the context of operational calculus [473], discrete analytic functions [474], partial differential equations [475–477], random walks [471, 478], and lattice Green's function [479]. Notable examples include the expressions by A. Stöhr [475],

$$R(x,y) = -\frac{1}{2\pi}\int_0^\infty \left[\left(1-\frac{t}{\zeta}\right)^{x+y}\left(1-\frac{t}{\zeta^3}\right)^{x-y}(1-\zeta t)^{-x+y}\left(1-\zeta^3 t\right)^{-x-y}\right]\frac{dt}{t}, \tag{5.65}$$

$$\zeta = e^{\frac{2\pi i}{8}}, \tag{5.66}$$

and

$$R(x,y) = -\frac{1}{\pi}\int_0^\pi \frac{\left[\lambda-\sqrt{\lambda^2-1}\right]^y \cos x\theta}{\sqrt{\lambda^2-1}}d\theta, \tag{5.67}$$

$$\lambda = 2-\cos\theta, \tag{5.68}$$

F. Spitzer [478],

$$R(x,y) = \frac{1}{8\pi^2}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi}\frac{1-\cos(x\alpha+y\beta)}{1-\frac{1}{2}(\cos\alpha+\cos\beta)}d\alpha\, d\beta, \tag{5.69}$$

B. van der Pol [473],

$$R(x, y) = \frac{1}{2\pi} \int_0^\infty \left[1 - \left(\frac{t+i}{t-i} \right)^{x+y} \left(\frac{t-1}{t+1} \right)^{|x-y|} \right] \frac{dt}{t}, \tag{5.70}$$

and W. H. McCrea and F. J. W. Whipple [471], later rediscovered by G. Venezian [476] and J. Cserti [479],

$$R(x, y) = \frac{1}{\pi} \int_0^\pi \frac{1 - e^{-x\mu} \cos y\lambda}{\sinh \mu} d\beta, \tag{5.71}$$

$$\cosh \mu + \cos \lambda = 2. \tag{5.72}$$

Expressions (5.65) to (5.71) describe uniform resistive lattices. Many practical VLSI grids are anisotropic, *i.e.*, the resistance along the horizontal dimension is not the same as the resistance along the vertical dimension. An expression for the resistance within an infinite anisotropic resistive grid is presented in [469],

$$R(x, y, k) = \frac{kr}{\pi} \int_0^\pi \frac{2 - e^{-|x|\alpha} \cos y\beta}{\sinh \alpha} d\beta, \tag{5.73}$$

where k is the ratio of the horizontal resistance to the vertical resistance, and

$$k + 1 = k \cos \beta + \cosh \alpha. \tag{5.74}$$

This result has significant value for the analysis of power grids. To determine the equivalent resistance within an $M \times N$ grid using MNA, a solution of the linear equation of size $MN \times MN$ is necessary, requiring prohibitive computational time. In contrast, the effective resistance between two nodes within a grid can be found in constant time, assuming these nodes are sufficiently far from the grid boundaries. A linear complexity, IR voltage drop analysis algorithm is introduced in [480]. The contribution of the voltage sources and current loads to IR voltage drops is evaluated separately based on the effective resistance computed in constant time. The solutions are superimposed to determine the total IR voltage drop within a circuit. The solution is further accelerated by observing that the IR voltage drop contribution of distant voltage sources and loads is negligible. By restricting the analysis to the vicinity of a node, the runtime can be drastically reduced while maintaining the error below 0.5%.

5.4 Summary

Due to the stringent performance requirements of modern VLSI systems, the demand for accurate circuit analysis has drastically increased over the past decades. The immense complexity of modern VLSI systems however makes standard circuit analysis based on MNA impractical. A wide range of algorithms have been proposed to reduce the runtime of the circuit analysis process while maintaining sufficient accuracy. The most prominent techniques are described in this chapter.

Domain decomposition methods split a circuit into multiple independent domains, thereby reducing the computational complexity and enabling parallelization. In the $\mathcal{H}$-matrix representation, the sparsity of practical matrices is exploited to produce a cluster tree, enabling efficient algorithms with less memory requirements and lower computational complexity. Using multigrid techniques, a solution is initially approximated using a coarse version of the system. The solution is subsequently determined after interpolation and smoothing operations.

Alternative circuit analysis techniques attempt to accelerate the circuit analysis process by avoiding costly MNA-based analysis. A complex or obfuscated circuit can be represented by a multiport network model, efficiently described by S parameters. In random walk-based methods, the voltage within a grid is determined statistically, yielding linear computational complexity at a fixed accuracy. The infinite lattice model can often be used to analyze large grids, often encountered in on-chip power distribution systems.

Common circuit analysis methods are discussed in this chapter. These methods enhance traditional MNA processes or follow alternative approaches. Despite the immense potential, few of these techniques are used in mainstream circuit analysis methodologies. Further research is required to improve the versatility and performance of the advanced circuit analysis techniques discussed in this chapter. For example, a significant limitation of infinite lattice analysis is poor accuracy near the boundaries of the grid. This limitation is overcome by applying the image method [71] and infinity mirror technique [72], described, respectively, in Chapters 6 and 7.