PiSPICE: Accelerating Post-Layout SPICE Simulation via Essential Parasitic Identification

Zhou Jin[1], Jing Li[2], Jian Xin[3], Tianjia Zhou[3], Xiao Wu[4], Dan Niu[5], and Zuochang Ye[3]

[1] College of Integrated Circuits, Zhejiang University, China
[2] SSSLab, China University of Petroleum-Beijing, China
[3] Tsinghua University, China
[4] Huada Empyrean Software Co. Ltd, China
[5] School of Automation, Southeast University, China
Email: z.jin@zju.edu.cn

62 THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY CEDA® IEEE Council on Electronic Design Automation sigda acm

OUTLINE

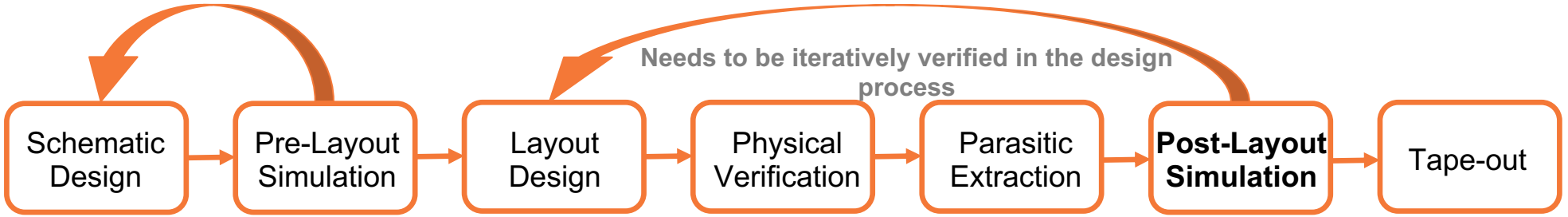
- Background
- Motivation
- PiSPICE
- Experiment
- Conclusions

Background

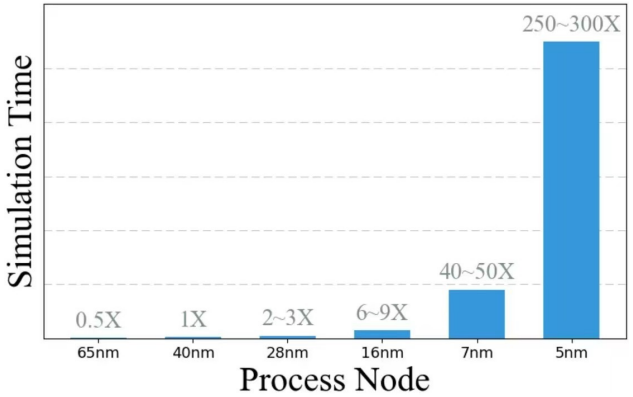
Background

- **Post-layout SPICE simulation** accurately models circuit behavior under real operating conditions by accounting for **parasitic effects** introduced by the physical layout and interconnects. It is essential for validating circuit performance, achieving timing signoff, and ensuring design yield and reliability.

Design process of analog circuits



- With advancing process nodes and tighter design margins, traditional post-layout SPICE simulations are becoming **computationally prohibitive**, both in terms of memory and processing time.



Growth of post-layout simulation time across different process nodes

Circuit	Pre-Layout Simulation Time	Post-Layout Simulation Time	Time Growth
DAC	1.1 h	139 h	126.4 x
ADC	1.3 h	273 h	210 x
Buck	0.4 h	521 h	1302.5 x
CODEC	89.4 h	-	-

Comparison of pre-layout and post-layout simulation time for the same circuit

Background

- The **Domain Decomposition Method (DDM)** partitions the circuit into smaller subdomains and organizes the system into a bordered block diagonal (BBD) matrix, thereby facilitating parallel computation and simplifying the overall solution process.

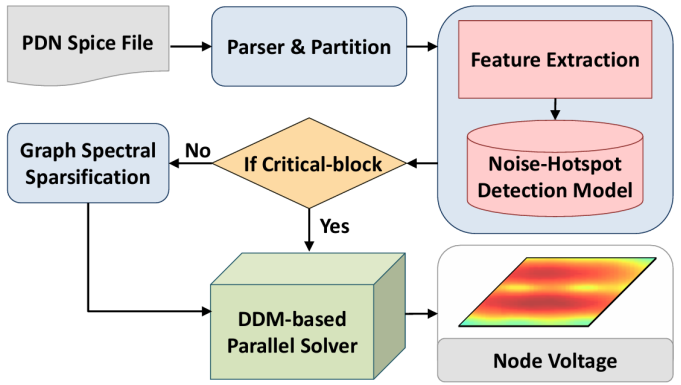
$$Gv = b$$

$$\begin{bmatrix} G_1 & E_{12} & \cdots & E_{1,m} \\ E_{12}^T & G_2 & \cdots & E_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ E_{1,m}^T & E_{2,m}^T & \cdots & G_m \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

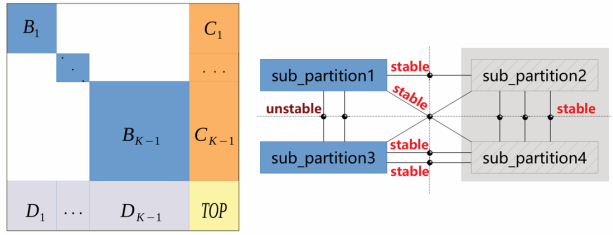
Subdomain $\begin{bmatrix} A_i & F_i \\ F_i^T & C_i \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} f_i \\ g_i \end{bmatrix}$

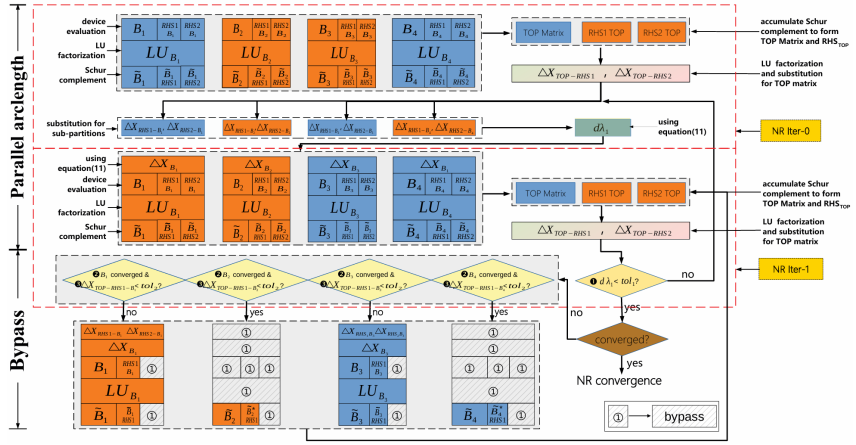
The approximated matrices for non-critical blocks

$$\begin{bmatrix} \tilde{G}_1 & E_{12} & \cdots & E_{1,m} \\ E_{12}^T & \tilde{G}_2 & \cdots & E_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ E_{1,m}^T & E_{2,m}^T & \cdots & \tilde{G}_m \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$



A Parallel Simulation Framework Incorporating Machine Learning-Based Hotspot Detection for Accelerated Power Grid Analysis [1]





PALBBD [2]

[1] Y. Jiang, J. Song, X. Yang, X. Dong, S. Sun, Y. Lin, Z. Jin, X. Yin and C. Zhuo, A Parallel Simulation Framework Incorporating Machine Learning-Based Hotspot Detection for Accelerated Power Grid Analysis. MLCAD, 2024.

[2] Z. Jin, T. Feng, Y. Duan, X. Wu, M. Cheng, Z. Zhou and W. Liu, PALBBD: A Parallel ArcLength Method Using Bordered Block Diagonal Form for DC Analysis. GLSVLSI, 2021.

Background

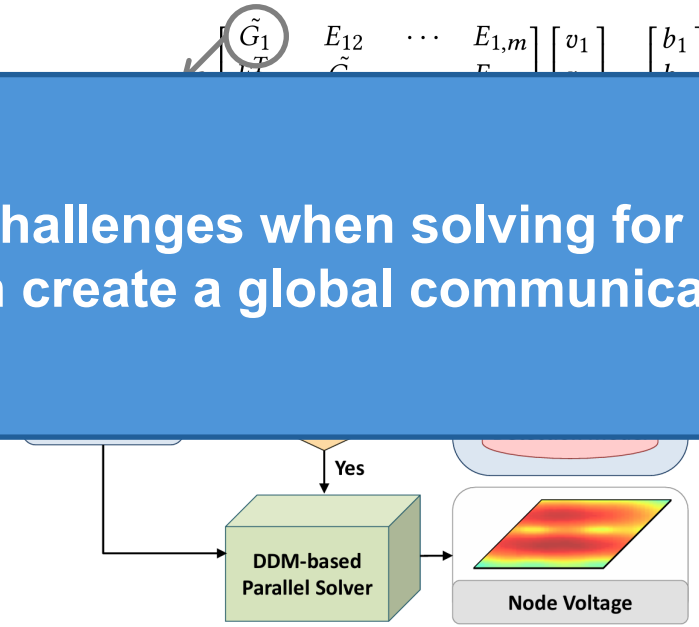
- The **Domain Decomposition Method (DDM)** partitions the circuit into smaller subdomains and organizes the system into a bordered block diagonal (BBD) matrix, thereby facilitating parallel computation and simplifying the overall solution process.

$$Gv = b$$

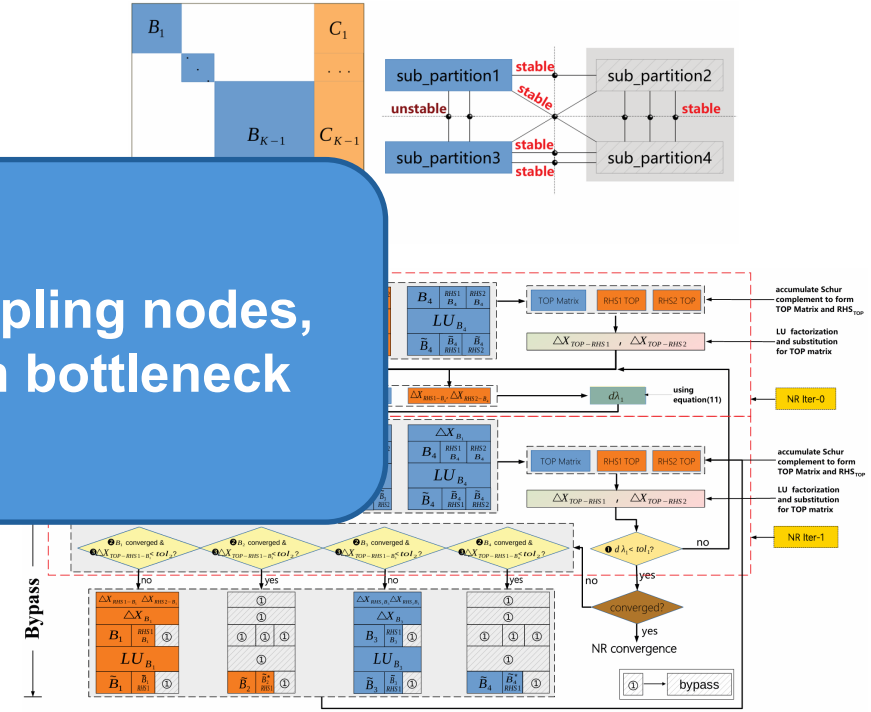
$$\begin{bmatrix} G_1 & E_{12} & \cdots & E_{1,m} \\ E_{12}^T & G_2 & \cdots & E_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ E_{1,m}^T & E_{2,m}^T & \cdots & G_m \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix}$$

$$\text{Subdomain} \quad \begin{bmatrix} A_i & F_i \\ F_i^T & C_i \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} f_i \\ g_i \end{bmatrix}$$

Face challenges when solving for coupling nodes, which create a global communication bottleneck



A Parallel Simulation Framework Incorporating Machine Learning-Based Hotspot Detection for Accelerated Power Grid Analysis [1]



PALBBD [2]

[1] Y. Jiang, J. Song, X. Yang, X. Dong, S. Sun, Y. Lin, Z. Jin, X. Yin and C. Zhuo, A Parallel Simulation Framework Incorporating Machine Learning-Based Hotspot Detection for Accelerated Power Grid Analysis. MLCAD, 2024.

[2] Z. Jin, T. Feng, Y. Duan, X. Wu, M. Cheng, Z. Zhou and W. Liu, PALBBD: A Parallel ArcLength Method Using Bordered Block Diagonal Form for DC Analysis. GLSVLSI, 2021.

Background

- **Model Order Reduction (MOR)** simplifies the system by approximating it with a lower-order model that retains the dominant behavior of the circuit.

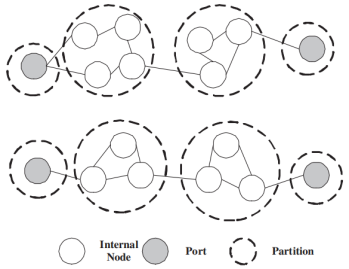
Algorithm 1 $[\hat{G},\hat{C},\hat{E}]$=SIPcore($G$, C, E, ports)

Get the optimal ordering to reduce the fill-in of matrix G
move port nodes to the bottom
permute matrices G and C
for $i = 1$ to $n - m$ **do**
  Construct matrix $M^{(i)}$
  $G^{(i+1)} \leftarrow {M^{(i)}}^T G^{(i)} M^{(i)}$
  $C^{(i+1)} \leftarrow {M^{(i)}}^T C^{(i)} M^{(i)}$
end for
$\hat{G} = G^{(n-m+1)}(n-m+1:n, n-m+1:n)$
$\hat{C} = C^{(n-m+1)}(n-m+1:n, n-m+1:n)$

Algorithm 2 $[\hat{G},\hat{C},\hat{E}]$=SIP($G$, C, E, ports, s_1,s_2,...,s_q)

for $i = 1$ to q **do**
  $[\hat{G}_i,\hat{C}_i]$=SIPcore($G + s_i C$, C, ports)
end for
for $i = 1$ to q **do**
  for $j = 1$ to i **do**
    $\hat{C}_{ji} = \frac{1}{s_j - s_i}(\hat{G}_j - \hat{G}_i)$
    $\hat{G}_{ji} = \hat{G}_i - s_i \hat{C}_{ji}$
  end for
end for

SIP [1]

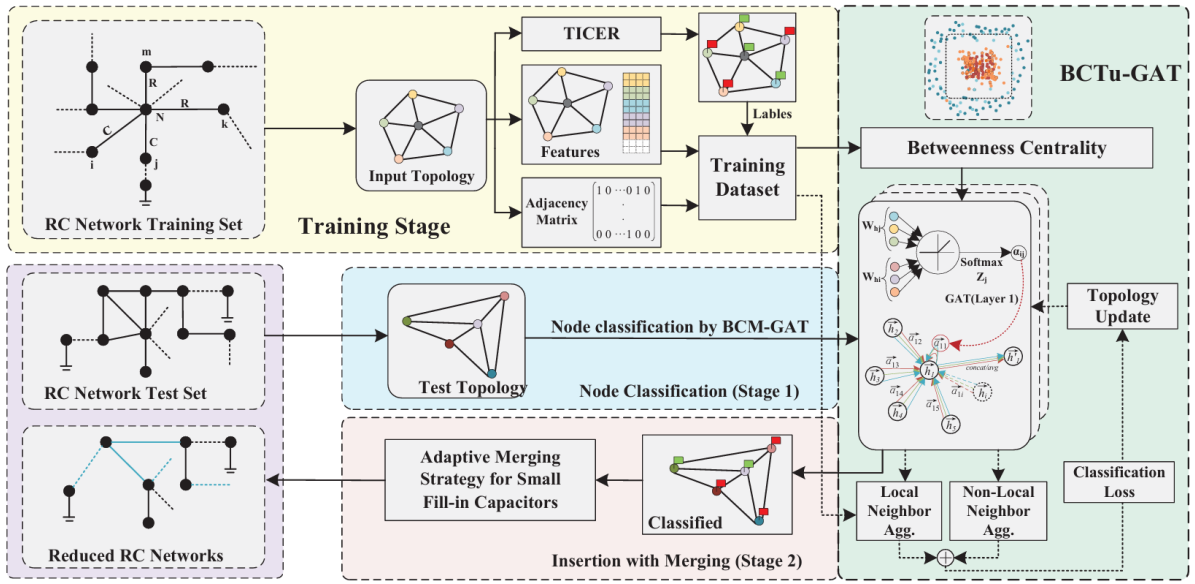


Algorithm 3 Aggregating Procedure

Input: the RC network, and the partition result obtained by spectral partition algorithm
Output: the reduced-order circuit

1: **for** Each Partition **do**
2: Use a "super node" to represent all the nodes in the partition.
3: Neglect the resistors and capacitors connected between the nodes in the partition.
4: Denote $\{r_1, r_2, \cdots, r_j\}$, and $\{c_1, c_2, \cdots, c_l\}$ the resistors and capacitors connected between the nodes in the partitions and ground. Add a resistor $r = 1/(\sum_{i=1}^{j} 1/r_i)$, and a capacitor $c = \sum_{i=1}^{l} c_i$ between the "super node" and ground.
5: **end for**
6: The capacitors and resistors connected between the nodes of different partitions are now connected between the "super nodes" which represent the partitions.
7: Use equivalent resistor/capacitor to represent the multiple resistors/capacitors parallel connected between the same pair of nodes.

AMOR [2]



TSA-TICER [3]

[1] Z. Ye, D. Vasilyev, Z. Zhu and J. Phillips, Sparse Implicit Projection (SIP) for Reduction of General Many-Terminal Networks. ICCAD, 2008.
[2] Y. Su, F. Yang and X. Zeng, AMOR: An Efficient Aggregating Based Model Order Reduction Method for Many-Terminal Interconnect Circuits. DAC, 2012.
[3] P. Chen, D. Niu, Z. Jin, C. Sun, Q. Li and H. Yan, TSA-TICER: A Two-Stage TICER Acceleration Framework for Model Order Reduction. DATE, 2024.

Background

- **Model Order Reduction (MOR)** simplifies the system by approximating it with a lower-order model that retains the dominant behavior of the circuit.

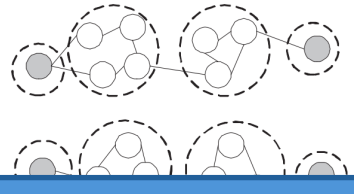
Algorithm 1 $[\hat{G},\hat{C},\hat{E}]$=SIPcore($G$, C, E, ports)

Get the optimal ordering to reduce the fill-in of matrix G
move port nodes to the bottom
permute matrices G and C
for $i = 1$ to $n - m$ **do**
  Construct matrix $M^{(i)}$
  $G^{(i+1)} \leftarrow {M^{(i)}}^T G^{(i)} M^{(i)}$
  $C^{(i+1)} \leftarrow {M^{(i)}}^T C^{(i)} M^{(i)}$
end for
$\hat{G} = G^{(n-m+1)}(n-m+1:n, n-m+1:n)$
$\hat{C} = C^{(n-m+1)}(n-m+1:n, n-m+1:n)$

Algorithm 2 $[\hat{G},\hat{C},\hat{E}]$=SIP($G$, C, E, ports, s_1,s_2,...,s_q)

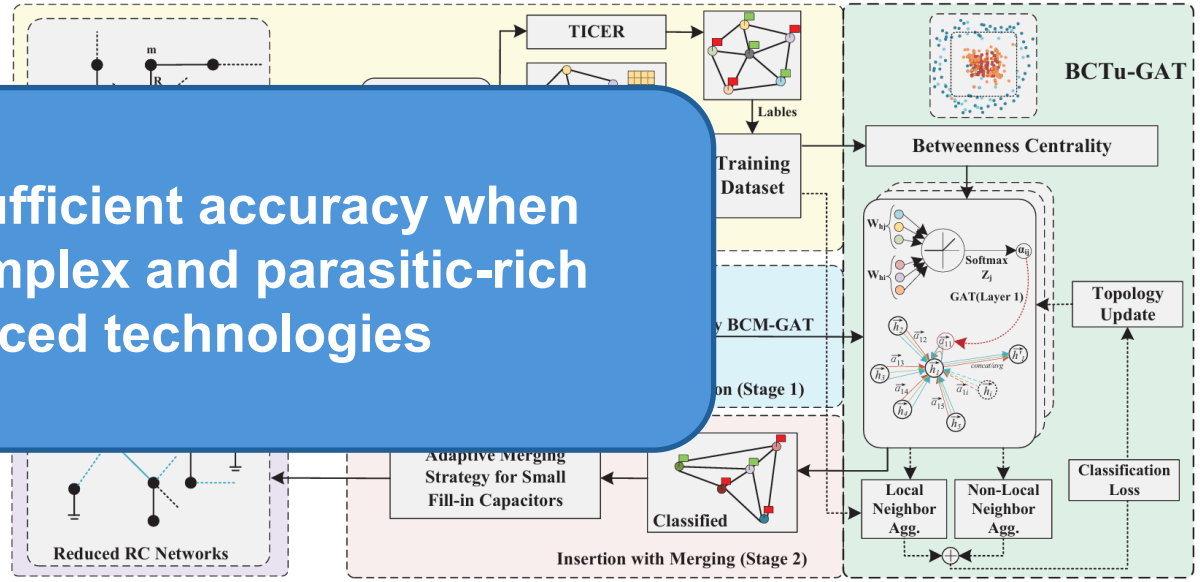
for $i = 1$ to q **do**
  $[\hat{G}_i,\hat{C}_i]$=SIPcore($G + s_i C$, C, ports)
end for
for $i = 1$ to q **do**
  for $j = 1$ to i **do**
    $\hat{C}_{ji} = \frac{1}{s_j - s_i}(\hat{G}_j - \hat{G}_i)$
    $\hat{G}_{ji} = \hat{G}_i - s_i \hat{C}_{ji}$
  end for
end for

SIP [1]



$1/(\sum_{i=1}^{j} 1/r_i)$, and a capacitor $c = \sum_{i=1}^{l} c_i$ between the "super node" and ground.
5: **end for**
6: The capacitors and resistors connected between the nodes of different partitions are now connected between the "super nodes" which represent the partitions.
7: Use equivalent resistor/capacitor to represent the multiple resistors/capacitors parallel connected between the same pair of nodes.

AMOR [2]



TSA-TICER [3]

Often fail to provide sufficient accuracy when dealing with highly complex and parasitic-rich circuits in advanced technologies

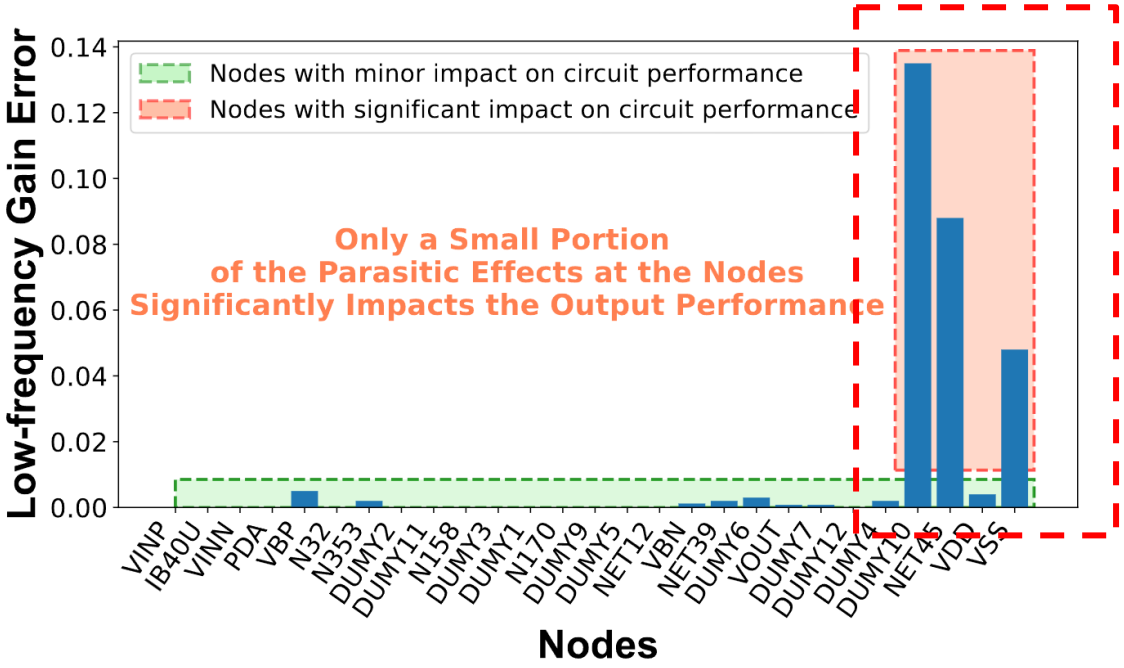
[1] Z. Ye, D. Vasilyev, Z. Zhu and J. Phillips, Sparse Implicit Projection (SIP) for Reduction of General Many-Terminal Networks. ICCAD, 2008.
[2] Y. Su, F. Yang and X. Zeng, AMOR: An Efficient Aggregating Based Model Order Reduction Method for Many-Terminal Interconnect Circuits. DAC, 2012.
[3] P. Chen, D. Niu, Z. Jin, C. Sun, Q. Li and H. Yan, TSA-TICER: A Two-Stage TICER Acceleration Framework for Model Order Reduction. DATE, 2024.

Motivation

SPONSORED BY CEDA® IEEE Council on Electronic Design Automation sigda acm

Motivation

- Existing approaches overlook the fact that **circuit performance is not equally sensitive to the variation of all nodes**.



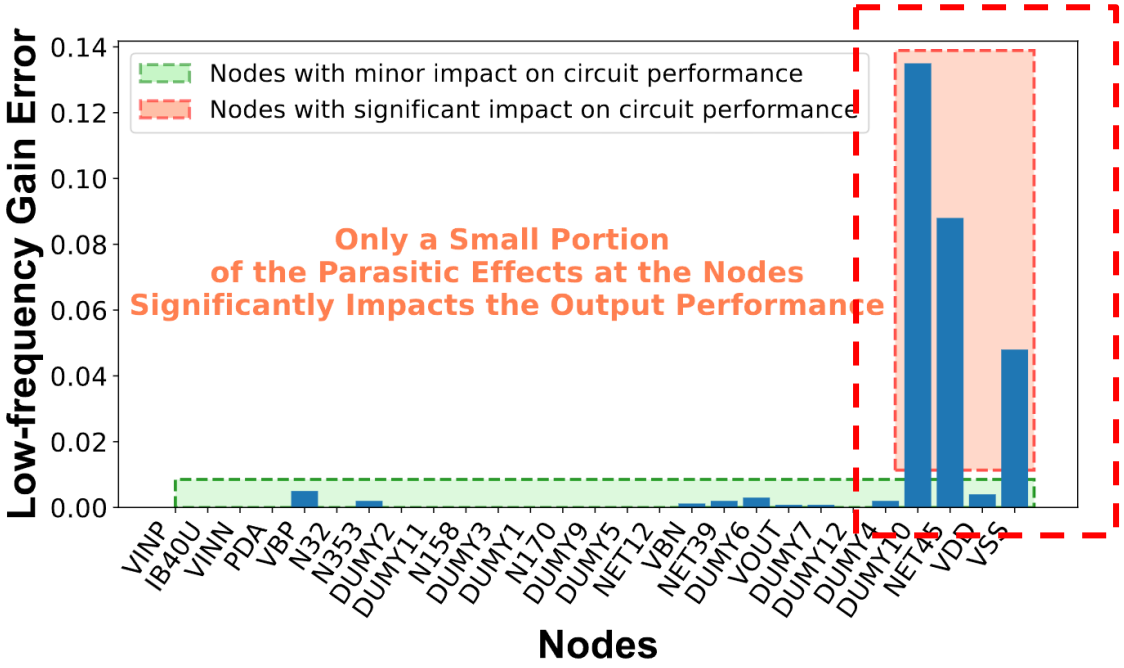
Low-frequency voltage gain error of an operational amplifier by removing parasitic sub-networks for each node.

Only a small portion of the parasitic effects of pre-layout node significantly affect circuit performance

The majority of parasitic sub-networks have **negligible** effects on the circuit performance

Motivation

- Existing approaches overlook the fact that **circuit performance is not equally sensitive to the variation of all nodes**.



Low-frequency voltage gain error of an operational amplifier by removing parasitic sub-networks for each node.

Only a small portion of the parasitic effects of pre-layout node significantly affect circuit performance

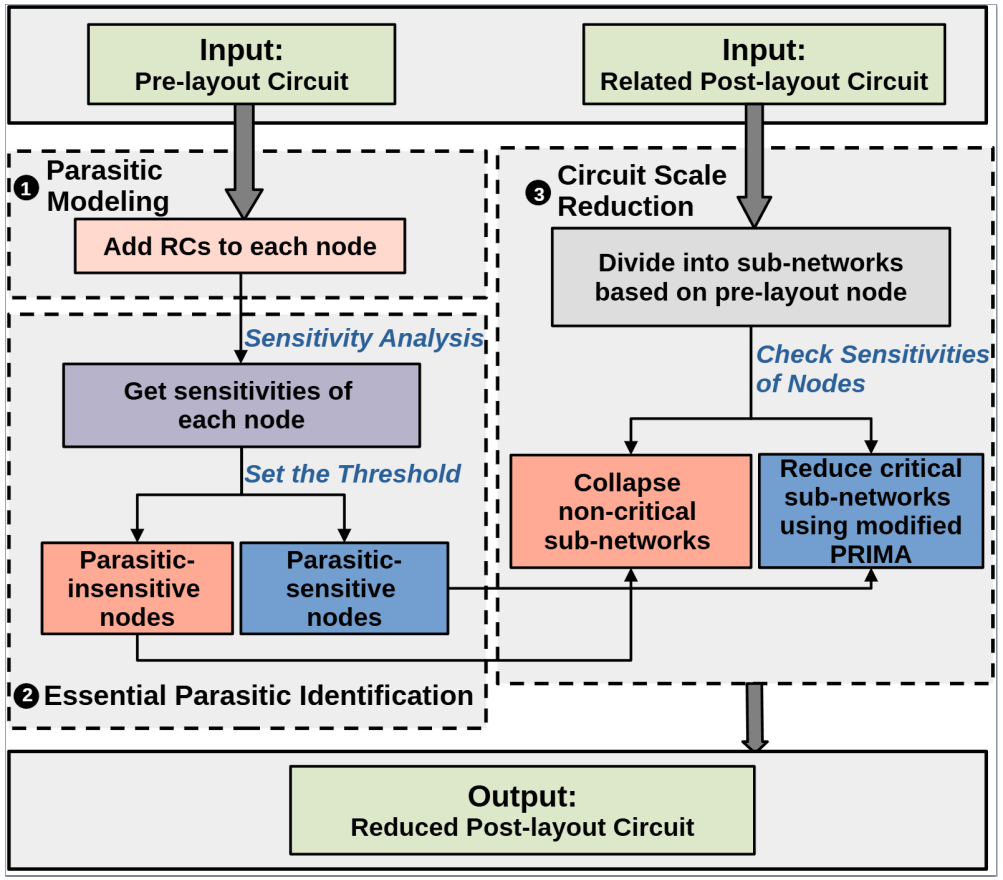
The majority of parasitic sub-networks have **negligible** effects on the circuit performance

Effectively identifying these parasitic-sensitive nodes and eliminating the less influential parasitic networks

PiSPICE

SPONSORED BY

Our Work-PiSPICE

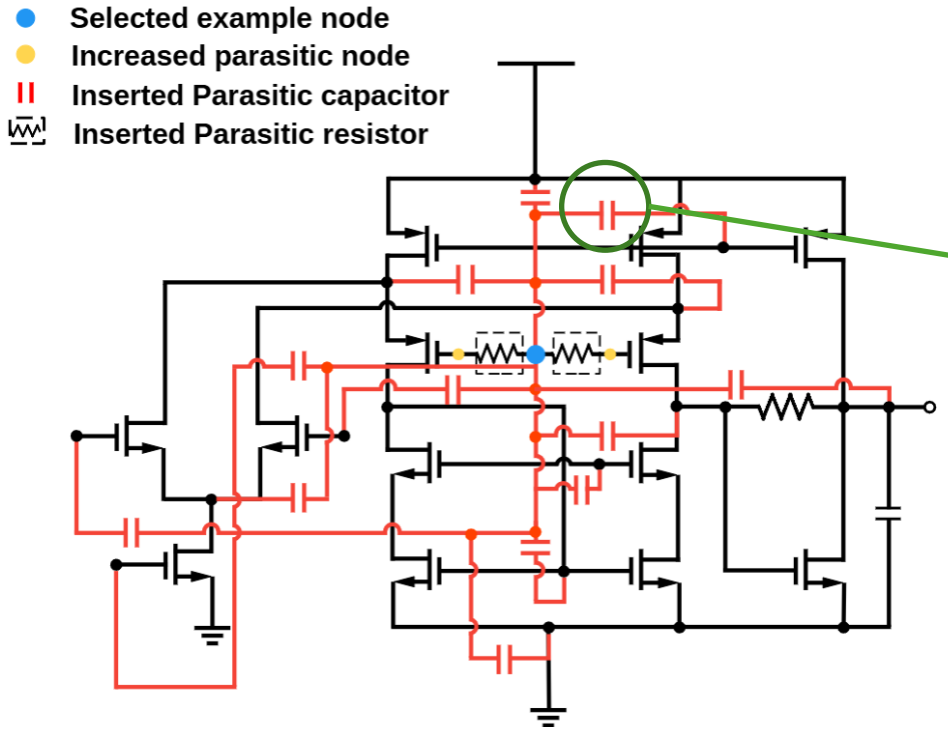


1. **Parasitic modeling** on the pre-layout circuit

2. Perform adjoint sensitivity analysis in the modeled circuit to **identify** parasitic-sensitive pre-layout nodes

3. **Reduce the RC networks** according to the sensitivity results

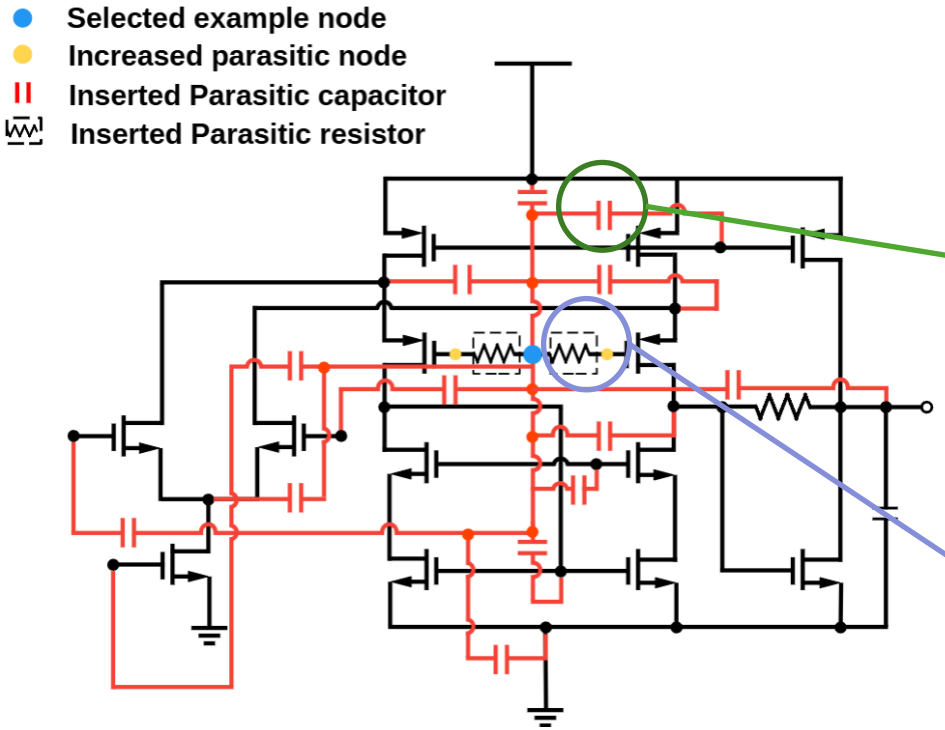
STEP1: Parasitic Modeling on the Pre-layout Circuit



Parasitic modeling for the selected blue node

- **Intrinsic Capacitance**
  Insert a capacitor between each node and the ground (VSS)
- **Coupling Capacitance**
  Insert a capacitor between each node and all other nodes

STEP1: Parasitic Modeling on the Pre-layout Circuit

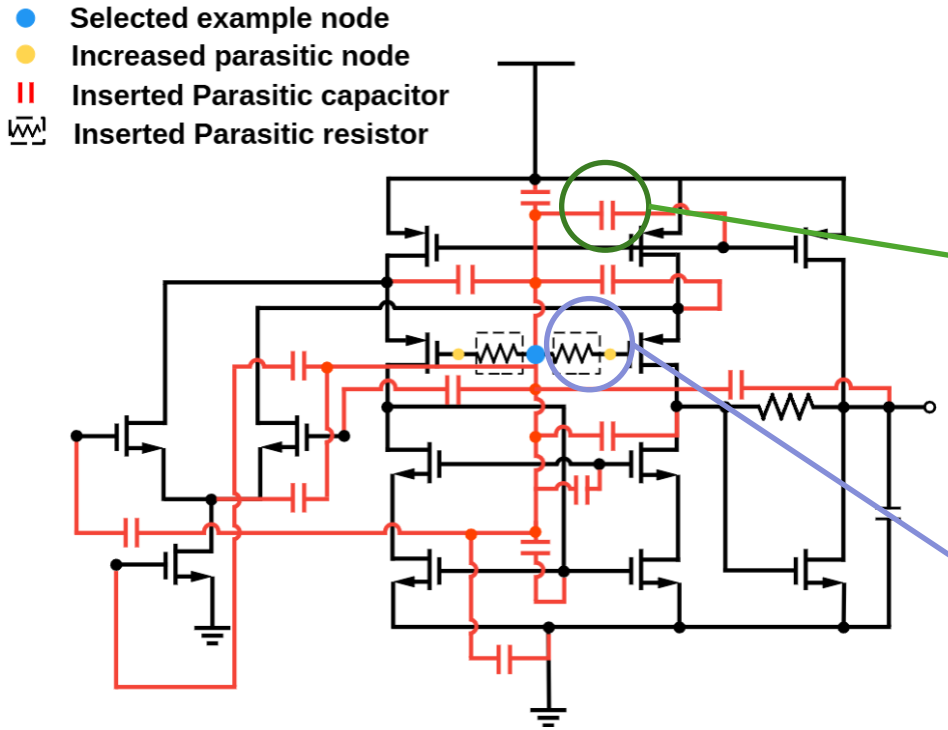


Parasitic modeling for the selected blue node

- **Intrinsic Capacitance**
  Insert a capacitor between each node and the ground (VSS)
- **Coupling Capacitance**
  Insert a capacitor between each node and all other nodes

- **Resistance**
  Add a new node between the original node and its connected component, then insert a resistor between these two nodes

STEP1: Parasitic Modeling on the Pre-layout Circuit



Parasitic modeling for the selected blue node

- **Intrinsic Capacitance**
  Insert a capacitor between each node and the ground (VSS)
- **Coupling Capacitance**
  Insert a capacitor between each node and all other nodes

- **Resistance**
  Add a new node between the original node and its connected component, then insert a resistor between these two nodes

Identical parasitic values that are effectively close to zero

Ensure the impact of parasitic components is determined mainly by their topological placement rather than their parameter values

STEP2: Essential Parasitic Identification

- **Sensitivity analysis** is performed by calculating the derivative of the objective function *f* (***p***) with respect to the parameters ***p*** = {p_1, p_2, ..., p_n}.

$$\frac{\partial f}{\partial p_i}$$ → Circuit performance; → Each parasitic parameter

- **Adjoint sensitivity analysis** calculates sensitivities with respect to multiple parameters simultaneously by solving the adjoint equation, requiring **significantly fewer simulations** than traditional approach.

Time domain

$$f(x,t,p) = \frac{d}{dt}[Q(x,p)] + F(x,p) + B(t,p) = 0$$

$$\frac{dO}{dp} = \frac{dO}{dx}\frac{\partial x}{\partial p} = -\sum_{n=0}^{N} \lambda_n \left(\frac{\partial g}{\partial p}\right)_n$$

$$\lambda_n = \frac{1}{\delta_t}\lambda_{n+1}\left(\frac{\partial Q}{\partial x}\right)_n \left(\frac{\partial f}{\partial x}\right)_n^{-1}$$

$$\lambda_N = \frac{dO}{dx}\left(\frac{\partial f}{\partial x}\right)_N^{-1}$$

$$\left(\frac{\partial g}{\partial p}\right)_n = \frac{1}{\delta_t}\left[\left(\frac{\partial Q}{\partial p}\right)_n - \left(\frac{\partial Q}{\partial p}\right)_{n-1}\right] + \left(\frac{\partial F}{\partial p}\right)_n + \left(\frac{\partial B}{\partial p}\right)$$

Frequency domain

$$G(p)\,\Delta x(t) + C(p)\frac{d\,\Delta x(t)}{dt} + \Delta u = 0$$

$$G(p)Xe^{j\omega t} + C(p)\frac{d(Xe^{j\omega t})}{dt} + Ue^{j\omega t} = 0$$

$$G(p)Xe^{j\omega t} + j\omega C(p)d(Xe^{j\omega t}) + Ue^{j\omega t} = 0$$

$$G(p)X + j\omega C(p)X + U = 0$$

$$X = -\frac{U}{G(p)+j\omega C(p)}$$

$$\frac{\partial X}{\partial p} = U\frac{\frac{\partial G}{\partial p}+j\omega\frac{\partial C}{\partial p}}{(G+j\omega C)^2}$$

STEP2: Essential Parasitic Identification

- Calculate the **parasitic sensitivity for each pre-layout node**.

$$S_{\text{node},j} = \frac{1}{|\mathcal{N}_j|} \sum_{i \in \mathcal{N}_j} \left| \frac{\partial f}{\partial p_i} \right|$$

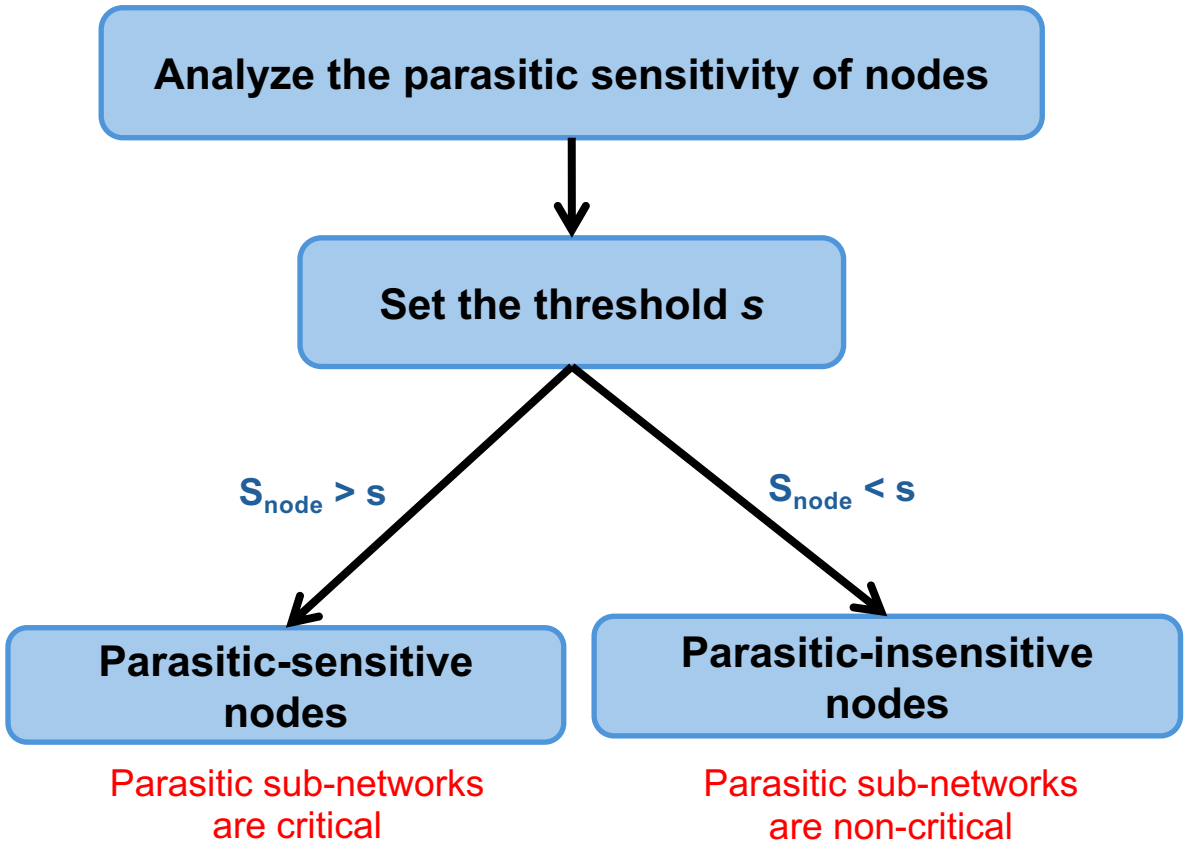
The number of parasitic RCs connected to the node

- Establish a fair **threshold** to **distinguish** between parasitic-sensitive and parasitic-insensitive nodes.

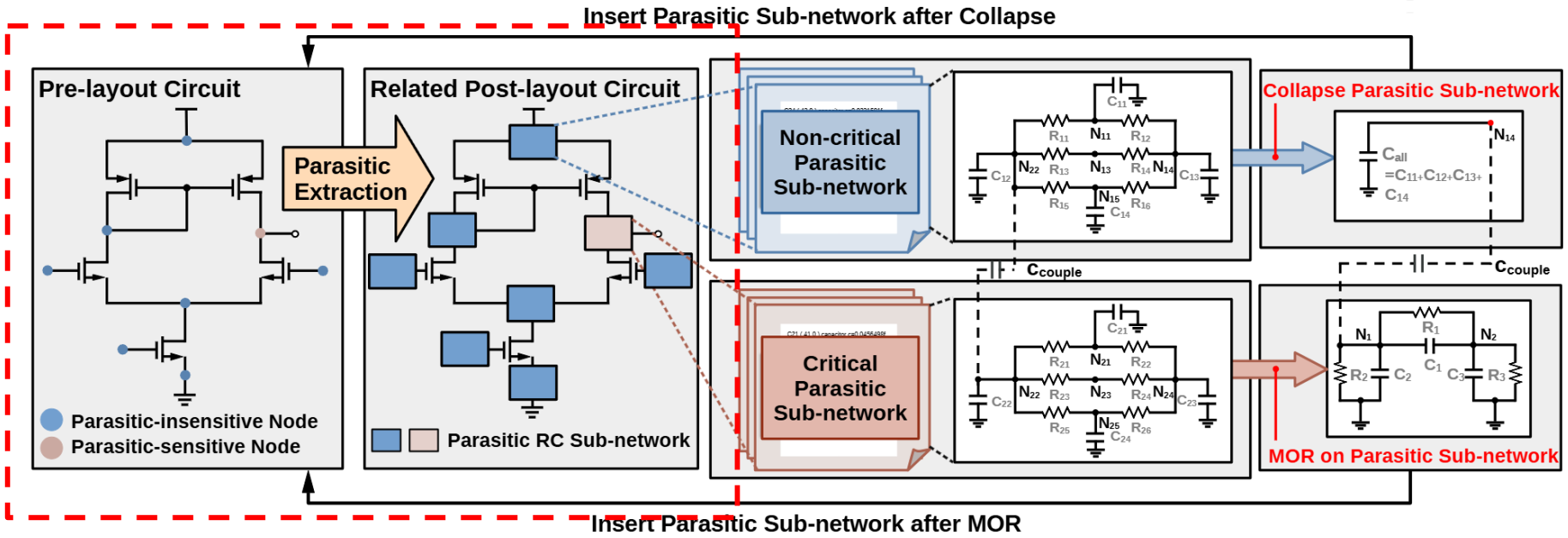
$$S_{\text{avg}} = \frac{1}{N} \sum_{j=1}^{N} S_{\text{node},j}$$

The number of pre-layout node

A user-adjustable safety factor s_u is introduced to balance the relaxation of sensitivity judgment between desired accuracy and speed



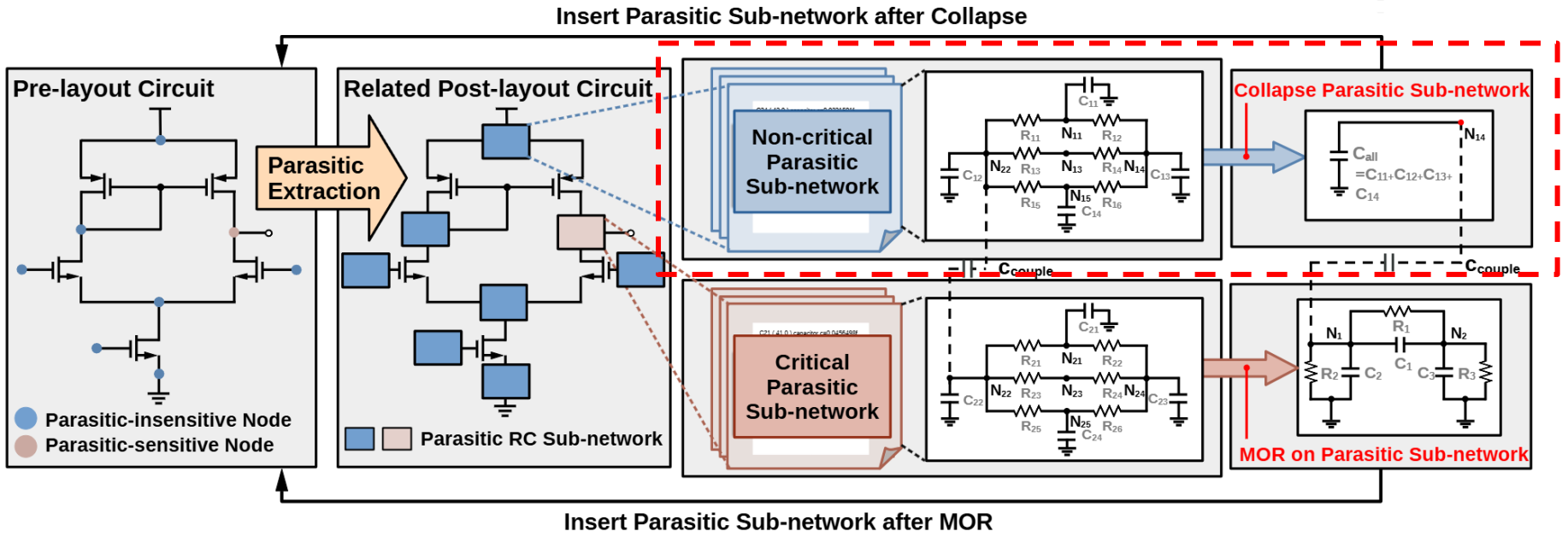
STEP3: Hybrid Strategy to Reduce Simulation Scale



- **Partition the Circuit**

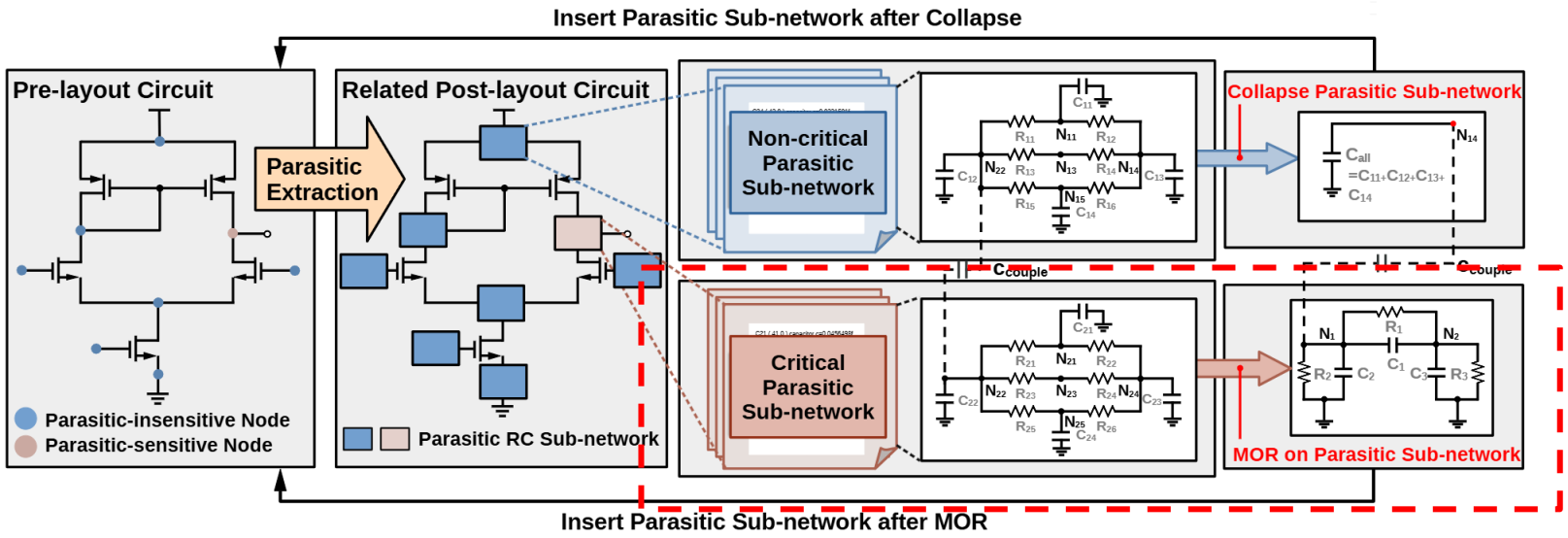
  Post-layout nodes originating from the same pre-layout node are grouped within the same sub-network

STEP3: Hybrid Strategy to Reduce Simulation Scale



- **Collapse the parasitic sub-network for nodes that are parasitic-insensitive**
  - All resistors are treated as short-circuit
  - All capacitors are treated as open-circuit
  - All ground capacitors are replaced by an equivalent capacitor
  - All nodes are merged into a single node

STEP3: Hybrid Strategy to Reduce Simulation Scale



New projection matrix

$$G^{-1}B = \begin{pmatrix} -A^{-1}B \\ I \end{pmatrix}$$

➢ **Reduce the parasitic sub-network for nodes that are parasitic-sensitive using improved PRIMA**

- Use the projection matrix $R = G^{-1}B$

Matrix reordering

$$G = \begin{pmatrix} A & B \\ B^T & D \end{pmatrix}$$

A—Composed of port nodes
B—Composed of ports to internal nodes
D—Composed of internal nodes

Port nodes: Nodes connected to one end of a transistor
Internal nodes: Other nodes

The ***G*** matrix of a sub-network is rank-deficient ! ✖

Experiment

Experimental Setup

➢ Experimental platform

- System equipped with an AMD EPYC 2.3 GHz CPU and 377 GB of memory

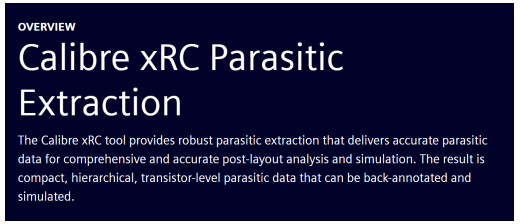
➢ Test Cases

Node counts ranging from 1k to 60k

- CKT1-CKT6 are operational amplifier circuits of varying scales
- CKT7 is a bandgap reference circuit
- CKT8 is a 4-bit SAR ADC circuit
- CKT9 is an ultrafast clock fan-out buffer circuit

➢ Tools

- **Parasitic extraction:** Calibre xRC
- **Simulator:** Cadence Spectre



Reduction and Acceleration Efficiency

Node reduction and acceleration efficiency comparisons with different methods.

Circuit	#Nodes			Time (ms)			PiSPICE vs Original		PiSPICE vs TICER		Relative Error
	Original	TICER	PiSPICE	Original	TICER	PiSPICE	Reduction Rate	Speedup	Reduction Rate	Speedup	
CKT1 (Ac)	1219	567	16	135.699	40.176	19.686	76.19x	6.89x	35.44x	2.04x	0.05%
CKT1 (Tran)	1219	567	13	189.536	119.603	34.802	**93.77x**	5.45x	**43.62x**	3.44x	0.30%
CKT2 (Ac)	3457	1623	235	616.297	137.021	81.820	14.71x	7.53x	6.91x	1.67x	0.05%
CKT2 (Tran)	3457	1623	568	640.569	261.536	106.922	6.09x	5.99x	2.86x	2.45x	0.16%
CKT3 (Ac)	5048	2447	422	541.906	136.357	53.167	11.96x	10.19x	5.80x	2.56x	0.18%
CKT3 (Tran)	5048	2447	1018	926.216	408.313	140.629	4.96x	6.59x	2.40x	2.90x	0.08%
CKT4 (Ac)	7194	3370	987	3397.520	1159.860	196.770	7.29x	**17.27x**	3.41x	5.89x	0.37%
CKT5 (Ac)	8920	4272	1687	3025.370	733.360	280.880	5.29x	10.77x	2.53x	2.61x	0.34%
CKT6 (Ac)	12255	6329	455	2052.257	447.222	227.815	26.93x	9.01x	13.91x	1.96x	0.25%
CKT7 (Ac)	24306	12743	2171	6430.030	2753.240	888.470	11.20x	7.24x	5.87x	3.10x	0.43%
CKT8 (Tran)	46894	27288	23768	9021930.000	5040080.000	4380000.000	1.97x	2.06x	1.15x	1.15x	**0.78%**
CKT9 (Tran)	68450	37546	29860	10800002.043	6453020.000	679281.022	2.29x	15.90x	1.26x	**9.50x**	0.56%
Average		-			-		21.89x	8.74x	10.43x	3.27x	0.30%

- **Compared with Original:** Node reduction ratio of PiSPICE can reach a maximum of **93.77x**, and an average of **21.89x**.
- **Compared with TICER:** Node reduction ratio of PiSPICE can reach a maximum of **43.62x**, and an average of **10.43x**.

Reduction and Acceleration Efficiency

Node reduction and acceleration efficiency comparisons with different methods.

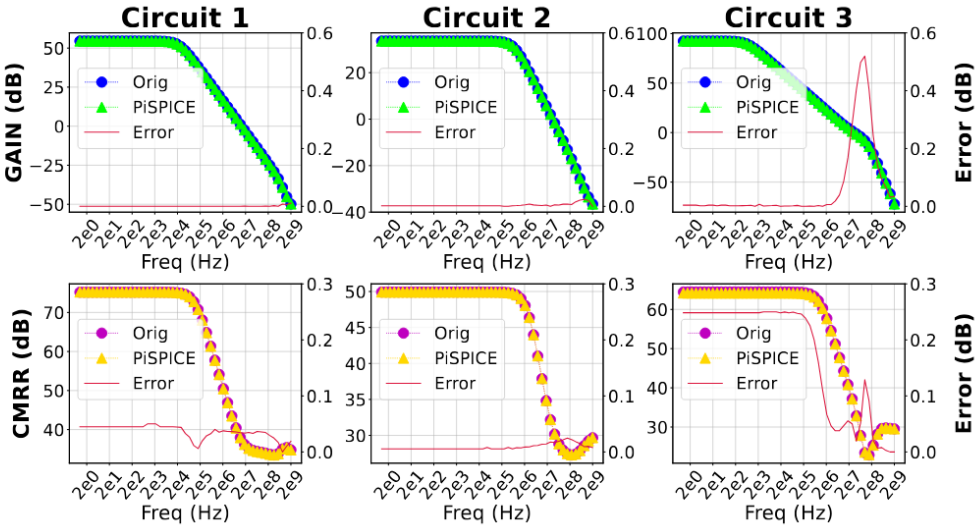
Circuit	#Nodes			Time (ms)			PiSPICE vs Original		PiSPICE vs TICER		Relative Error
	Original	TICER	PiSPICE	Original	TICER	PiSPICE	Reduction Rate	Speedup	Reduction Rate	Speedup	
CKT1 (Ac)	1219	567	16	135.699	40.176	19.686	76.19x	6.89x	35.44x	2.04x	0.05%
CKT1 (Tran)	1219	567	13	189.536	119.603	34.802	**93.77x**	5.45x	**43.62x**	3.44x	0.30%
CKT2 (Ac)	3457	1623	235	616.297	137.021	81.820	14.71x	7.53x	6.91x	1.67x	0.05%
CKT2 (Tran)	3457	1623	568	640.569	261.536	106.922	6.09x	5.99x	2.86x	2.45x	0.16%
CKT3 (Ac)	5048	2447	422	541.906	136.357	53.167	11.96x	10.19x	5.80x	2.56x	0.18%
CKT3 (Tran)	5048	2447	1018	926.216	408.313	140.629	4.96x	6.59x	2.40x	2.90x	0.08%
CKT4 (Ac)	7194	3370	987	3397.520	1159.860	196.770	7.29x	**17.27x**	3.41x	5.89x	0.37%
CKT5 (Ac)	8920	4272	1687	3025.370	733.360	280.880	5.29x	10.77x	2.53x	2.61x	0.34%
CKT6 (Ac)	12255	6329	455	2052.257	447.222	227.815	26.93x	9.01x	13.91x	1.96x	0.25%
CKT7 (Ac)	24306	12743	2171	6430.030	2753.240	888.470	11.20x	7.24x	5.87x	3.10x	0.43%
CKT8 (Tran)	46894	27288	23768	9021930.000	5040080.000	4380000.000	1.97x	2.06x	1.15x	1.15x	**0.78%**
CKT9 (Tran)	68450	37546	29860	10800002.043	6453020.000	679281.022	2.29x	15.90x	1.26x	**9.50x**	0.56%
Average		-			-		21.89x	8.74x	10.43x	3.27x	0.30%

- **Compared with Original:** Simulation acceleration ratio of PiSPICE can reach a maximum of **17.27x**, and an average of **8.74x**.
- **Compared with TICER:** Simulation acceleration ratio of PiSPICE can reach a maximum of **9.50x**, and an average of **3.27x**.

Accuracy Analysis

Node reduction and acceleration efficiency comparisons with different methods

Circuit	#Nodes			Time (ms)			PiSPICE vs Original		PiSPICE vs TICER		Relative Error
	Original	TICER	PiSPICE	Original	TICER	PiSPICE	Reduction Rate	Speedup	Reduction Rate	Speedup	
CKT1 (Ac)	1219	567	16	135.699	40.176	19.686	76.19x	6.89x	35.44x	2.04x	0.05%
CKT1 (Tran)	1219	567	13	189.536	119.603	34.802	**93.77x**	5.45x	**43.62x**	3.44x	0.30%
CKT2 (Ac)	3457	1623	235	616.297	137.021	81.820	14.71x	7.53x	6.91x	1.67x	0.05%
CKT2 (Tran)	3457	1623	568	640.569	261.536	106.922	6.09x	5.99x	2.86x	2.45x	0.16%
CKT3 (Ac)	5048	2447	422	541.906	136.357	53.167	11.96x	10.19x	5.80x	2.56x	0.18%
CKT3 (Tran)	5048	2447	1018	926.216	408.313	140.629	4.96x	6.59x	2.40x	2.90x	0.08%
CKT4 (Ac)	7194	3370	987	3397.520	1159.860	196.770	7.29x	**17.27x**	3.41x	5.89x	0.37%
CKT5 (Ac)	8920	4272	1687	3025.370	733.360	280.880	5.29x	10.77x	2.53x	2.61x	0.34%
CKT6 (Ac)	12255	6329	455	2052.257	447.222	227.815	26.93x	9.01x	13.91x	1.96x	0.25%
CKT7 (Ac)	24306	12743	2171	6430.030	2753.240	888.470	11.20x	7.24x	5.87x	3.10x	0.43%
CKT8 (Tran)	46894	27288	23768	9021930.000	5040080.000	4380000.000	1.97x	2.06x	1.15x	1.15x	**0.78%**
CKT9 (Tran)	68450	37546	29860	10800002.043	6453020.000	679281.022	2.29x	15.90x	1.26x	**9.50x**	0.56%
Average		-			-		21.89x	8.74x	10.43x	3.27x	0.30%



The gain curves, CMRR curves and absolute error in CKT1, CKT2 and CKT3 under different methods

- The maximum relative error introduced by PiSPICE remains below **0.78%**.
- The maximum absolute error introduced by PiSPICE is less than **0.6dB**.

Accuracy Analysis

User-adjustable safety factor s_u

- s_u = 0.5: Default threshold (baseline configuration)
- s_u = 0: Conservative mode – treats all nodes as parasitic-sensitive, preserving the original post-layout circuit (no reduction)
- s_u = 1: Aggressive mode – treats all nodes as parasitic-insensitive, eliminating all parasitic sub-networks (equivalent to pre-layout circuit)

The circuit scale and AC simulation accuracy of CKT3 after adjusting s_u

Circuit	Value of s_u	#Nodes	Time (ms)	Relative Error
CKT3 (Ac)	1.0	10	36.793	3.10%
	0.8	196	48.980	1.10%
	0.5	**422**	**53.167**	**0.18%**
	0.2	2747	336.357	0.06%
	0.0	5048	541.906	0.00%

By adjusting s_u, users can balance the relaxation of sensitivity judgment between desired accuracy and speed

Applicability Analysis

Performance of PiSPICE in post-layout simulation
with the same topology but different transistor parameters

Circuit	#Nodes		Time (ms)			Speedup	Relative Error
	Original	PiSPICE	Original	PiSPICE			
CKT3_1 (Ac)	5048	422	541.906	53.167	**Sensitivity**	10.19x	0.18%
CKT3_2 (Ac)	3955	157	287.349	37.225	**Analysis**	7.72x	0.26%
CKT3_3 (Ac)	4699	976	323.481	50.390	**Time:**	6.42x	0.20%
CKT3_4 (Ac)	21262	2053	1805.040	199.978	**221.526**	9.03x	0.50%
Average	-		-			8.34x	0.29%
Time Sum	-		**2957.776**	**562.286**		5.26x	-

- PiSPICE achieves great performance in **scenarios where the topology is the same but the parameters differ**, such as in sizing.
- Perform sensitivity analysis only **once**.

Overhead of Sensitive Analysis

Performance of PiSPICE in post-layout simulation
with the same topology but different transistor parameters

Circuit	#Nodes		Time (ms)			Speedup	Relative Error
	Original	PiSPICE	Original	PiSPICE			
CKT3_1 (Ac)	5048	422	541.906	53.167	**Sensitivity**	10.19x	0.18%
CKT3_2 (Ac)	3955	157	287.349	37.225	**Analysis**	7.72x	0.26%
CKT3_3 (Ac)	4699	976	323.481	50.390	**Time:**	6.42x	0.20%
CKT3_4 (Ac)	21262	2053	1805.040	199.978	**221.526**	9.03x	0.50%
Average	-		-			8.34x	0.29%
Time Sum	-		**2957.776**	**562.286**		5.26x	-

- The sensitivity analysis cost ranges from **12%** to **77%** of a full simulation run.
- Even including sensitivity analysis time, the total simulation time for four circuits achieves a **5.26x** speedup.

If more netlists that share the same topology need to be simulated during the sizing optimization process, the cost of sensitivity analysis becomes negligible

Conclusions

SPONSORED BY

Conclusions

In this paper, we propose PiSPICE, a novel framework designed to accelerate post-layout SPICE simulation by identifying and retaining only the essential parasitics.

- To our knowledge, this is the first work to accelerate post-layout simulation by identifying the significance of parasitics on circuit performance.
- Adjoint sensitivity analysis is performed on pre-layout circuit to identify essential parasitics, avoiding the costly computations of directly applying it to large post-layout circuits.
- The scale of the post-layout circuit is markedly reduced by collapsing non-critical parasitic sub-networks, and applying an improved PRIMA algorithm to critical parasitic sub-networks.
- PiSPICE is especially effective for circuits with identical topologies but varying parameters, requiring only one-time sensitivity analysis.
- PiSPICE achieves up to 93.77x circuit scale reduction and 17.27x speedup compared to Original, and 43.62x scale reduction with 9.50x speedup versus TICER, while maintaining simulation error below 0.78%.

AI

Security

Thanks for your listening !
Any qustions ?

Systems

EDA

Design

62 THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY CEDA® IEEE Council on Electronic Design Automation

sigda acm