SAND2012-4820P

Sparse Matrix Techniques for Next-Generation Parallel Transistor-level Circuit Simulation

Heidi K. Thornquist,
Siva Rajamanickam, Mike Heroux, and Erik Boman
Sandia National Laboratories

Parallel Matrix Algorithms and Applications
Birkbeck University of London, UK
June 28th, 2012

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

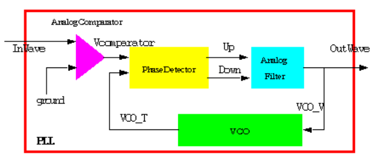
Outline

- Introduction
  - Analog circuit simulation
  - Transient simulation flow
- Impact of next-generation architectures
  - Device evaluations
  - Linear solvers
- Multi-core results
  - Hybrid linear algebra (Epetra)
  - Multithreaded device loads (Zoltan)
  - Hybrid-hybrid linear solver (ShyLU)
- Concluding Remarks

Simulation Hierarchy

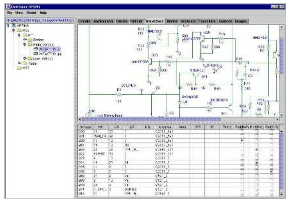
- Analog circuit simulation is just one of many types of simulation used by electrical designers.
- Tradeoff between fidelity and speed/problem size.
  - Digital simulation: **fast, low fidelity**
  - TCAD Device simulation: **slow, very high fidelity**
  - Circuit simulation: **in-between**

Speed ⟷ **Fidelity**

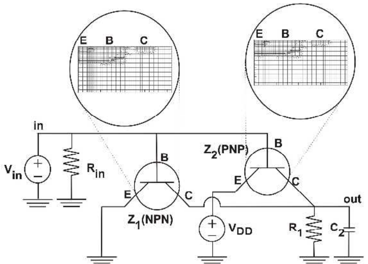


Digital (VHDL/Verilog)

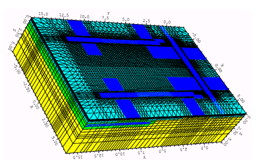
Mixed-Signal Habanero



Analog (SPICE)
Xyce



Mixed-Mode Xyce+Charon



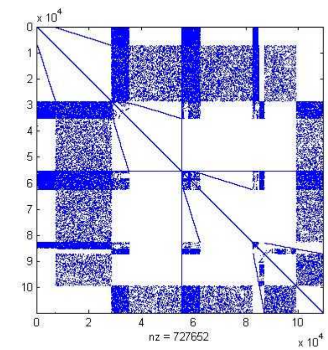
TCAD Device ***Charon***

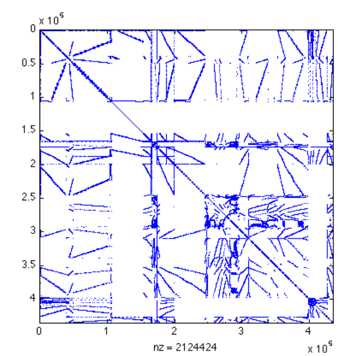
Parallel Circuit Simulation Challenges

Analog simulation models network(s) of devices coupled via Kirchoff's current and voltage laws

$$f(x(t)) + \frac{dq(x(t))}{dt} = b(t)$$

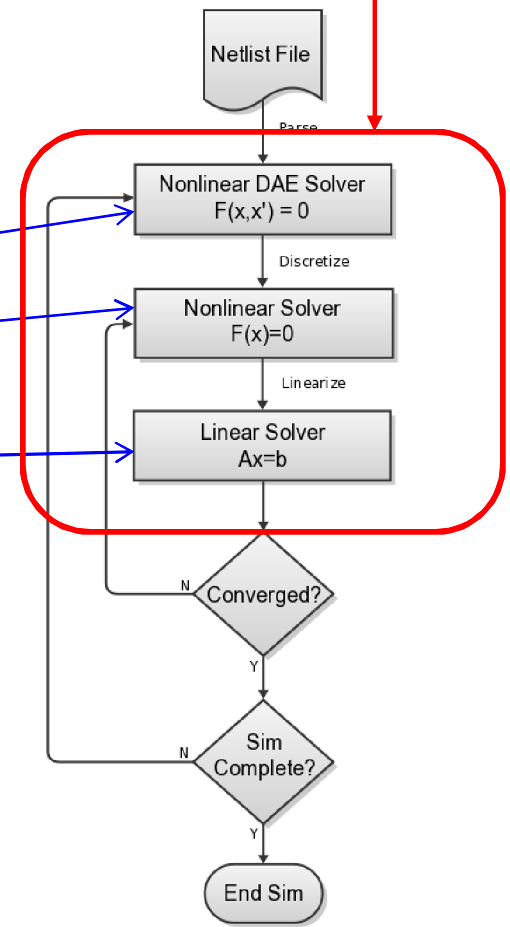
- Network Connectivity
  - Hierarchical structure rather than spatial topology
  - Densely connected nodes: O(n)
- Badly Scaled DAEs
  - Compact models designed by engineers, not numerical analysts!
  - Steady-state (DCOP) matrices are often ill-conditioned
- Non-Symmetric
  - Not elliptic and/or globally SPD
- Load Balancing / Partitioning
  - Balancing cost of loading Jacobian values unrelated to matrix partitioning for solves





Circuit Simulation Flow

Majority of simulation time spent here!



- Circuit simulators solve a system of nonlinear DAEs
  - How this is done depends on analysis type
  - Implicit integration methods
  - Newton's method
  - Sparse matrix techniques
- Transient simulation has□phases
  - Compute starting point (DCOP)
  - Start analysis (transient)
  - Sparse linear algebra / solvers
    - •Lynchpin of scalable performance

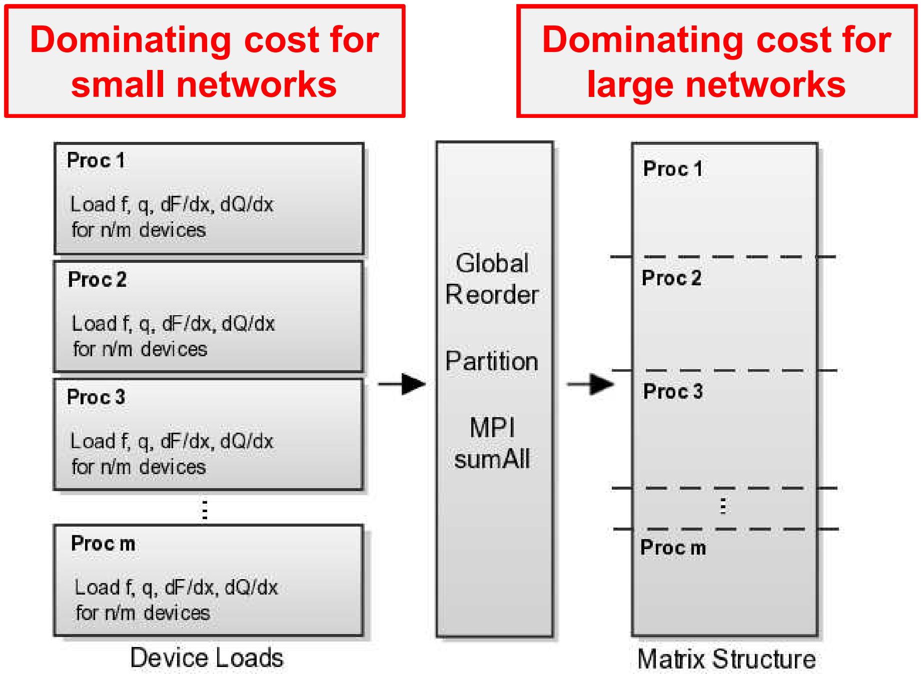
Sandia National Laboratories

Circuit Simulation Flow

- Multiple objectives for load balancing the solver loop
  - Device Loads : The partitioning of devices over processes will impact device evaluation and matrix loads
  - Matrix Structure : Graph structure is static throughout analysis, repartitioning matrix necessary for generating effective preconditioners

- Device Loads
  - Each device type can have a vastly different "cost" for evaluation
  - Memory for each device is considered separate
  - "Halo" exchanges may be very irregular

- Matrix Structure
  - Third-party libraries used to determine best graph structure and provide preconditioners / solvers

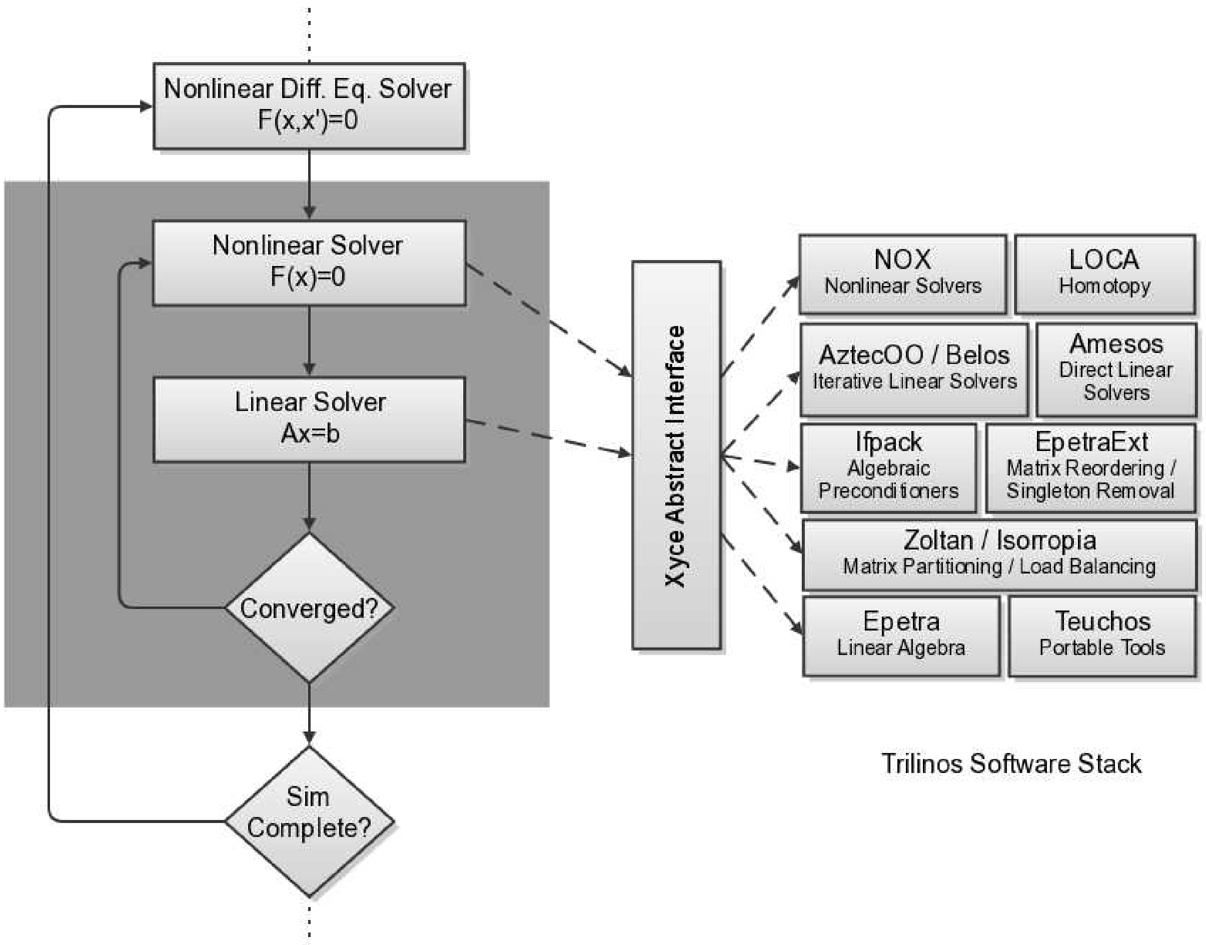


Sandia National Laboratories

Xyce™ Simulation Flow

PARALLEL ELECTRONIC SIMULATOR

Trilinos Software Stack



Trilinos Software Stack

- **Parallel linear algebra**
- **Advanced graph reordering and partitioning**
- **Preconditioners**
- **Parallel linear solvers**
- **Nonlinear methods (homotopy, continuation)**

Sandia National Laboratories

Impact of Next-Generation Architectures

- Requires a combination of programming paradigms / models
  - Analog circuit simulation has two dominant computational costs: device evaluation, linear solve
  - Re-evaluate simulation structure for intra-node parallelism
- Device Evaluation
  - Organize device evaluations for vectorization or threading (Zoltan)
  - Use computational kernels to address architecture differences (Kokkos)
- Linear Solve
  - Solvers that employ hybrid parallelism (ShyLU)
  - Linear algebra that takes advantage of node-level parallelism
    - Epetra (MPI / OpenMP)
    - Tpetra (Kokkos)

Multi-core Results

The results in this section are from simulations are performed on a small commodity cluster, where each node has a dual-socket/quad-core Intel Xeon® E5520 2.67 GHz processor and 36 GB of memory.

Epetra MPI / OpenMP Support

- First multi-core support for OpenMP in Trilinos 10.4
- Vector, MultiVector, CrsMatrix, CrsGraph support threading via OpenMP
- All computational methods are decorated with *parallel for* pragma's
- Use first-touch mechanism for optimal data placement (unless wrapping user data)
  - Can improve performance 2X on NUMA nodes

Epetra MPI only build (2 MPI procs)

Epetra hybrid build (2 MPI procs, 2 threads)

Circuit	Linear Solver (sec.)			Total Simulation (sec.)		
	MPI only	MPI w/OpenMP	x Speedup	MPI only	MPI w/OpenMP	x Speedup
ckt2	92.8	66.1	1.40	165.9	143.3	1.15
ckt3	246.7	101.2	2.43	351.0	198.4	1.76
ckt4	36.2	23.4	1.54	186.5	157.3	1.18
ckt5	92.9	46.3	2.00	239.5	181.3	1.32

Multithreaded Device Loads

- Devices are evaluated and loaded one device type at a time.
  - Evaluation is performed on independent memory.
  - Loading the Jacobian matrix and residual vector can result in race conditions.
- Multithreading the device loads is the challenge.
  - **Solution:** distance-1 coloring of the device sub-graph

A distance-1 coloring of $G = (V, E)$ is

- a mapping $\phi : V \rightarrow \{1, 2, \ldots, q\}$ s.t. $\phi(u) \neq \phi(v)$ whenever $(u, v) \in E$
- a partitioning of V into q independent sets

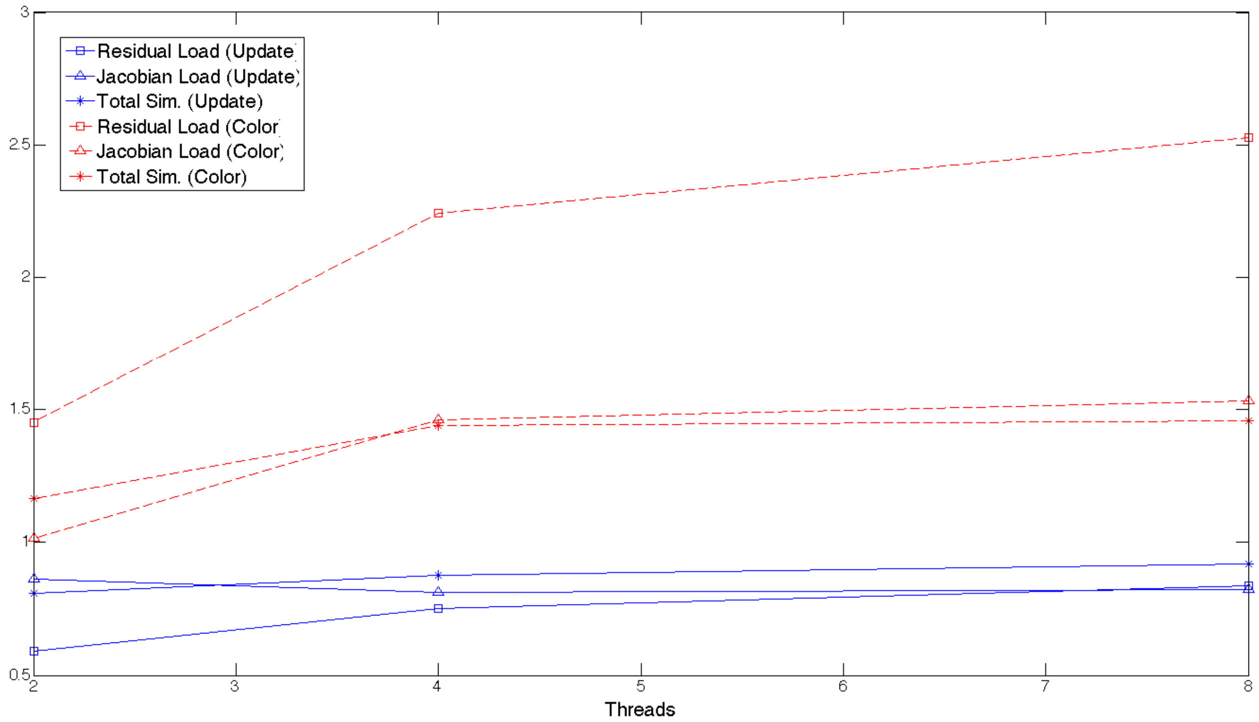
The objective is to minimize q

where V is the set of devices and E is the edges between those devices

- Colors define workgroups that can be loaded without race conditions.
- **Zoltan** is used to compute the distance-1 coloring of the device sub-graph.

Multithreaded Device Loads

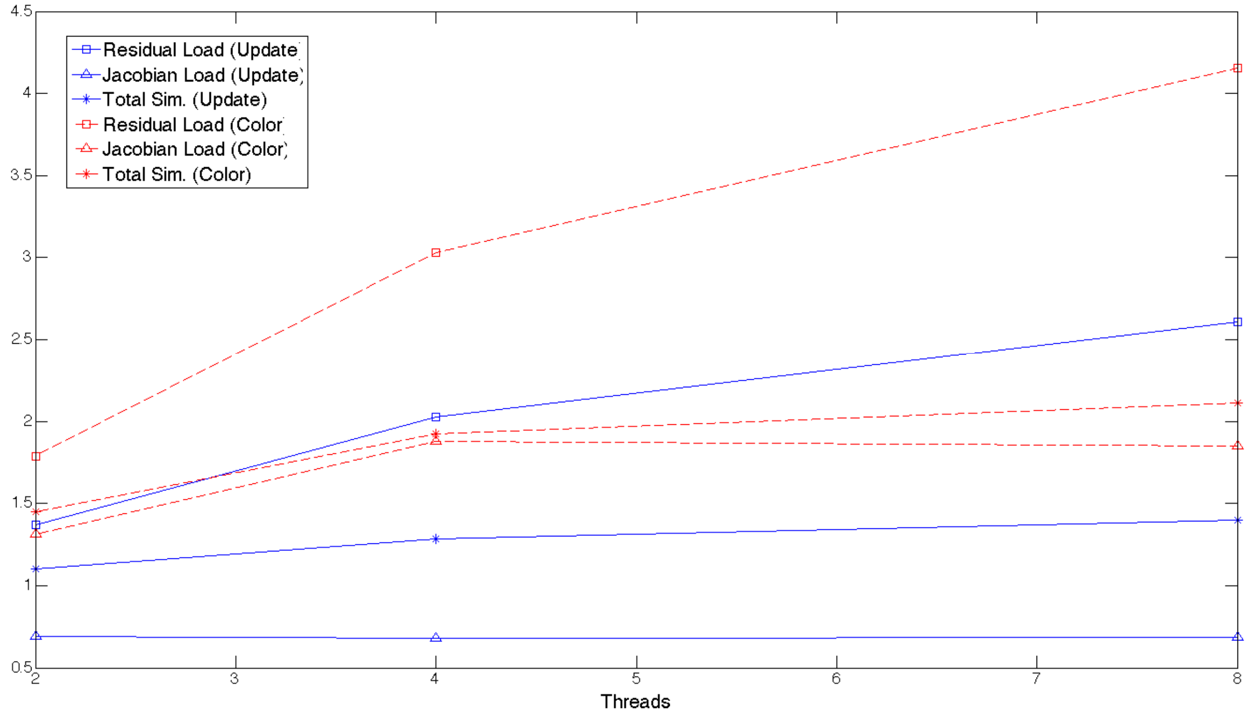
- Example 1: Simple 3k-stage RC ladder



- Linear solver = 33% serial simulation time
- Resistors (max vertex deg.=2; 2 colors), capacitors (max vertex deg.=1, 1 color)

Multithreaded Device Loads

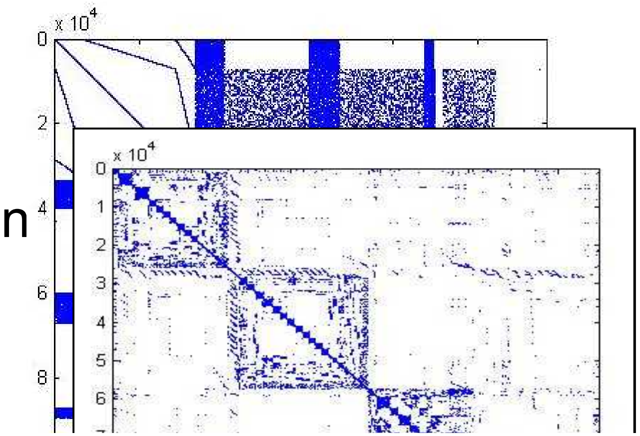
- Example 2: CMOS memory circuit (ckt5)

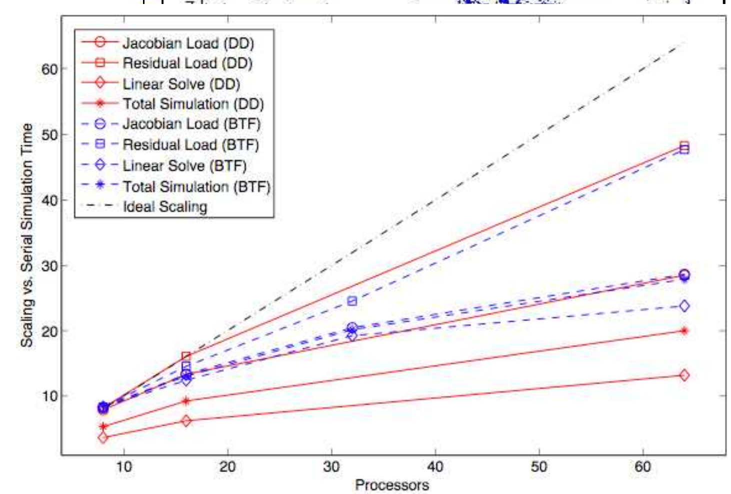


- Linear solver = 20% serial simulation time
- BSIM3 (max vertex degree 5098; no coloring) Resistor (max vertex degree 7; 7 colors) Voltage Source (max vertex degree 1; 1 color) Capacitor (max vertex degree 1; 1 color)

Linear Solver Scaling / Robustness

- Initially (circa 1999), Xyce used available PDE-based preconditioning techniques
  - Incomplete LU factorization
  - Limited scaling / robustness
- For small scale circuits, the Dulmage-Mendelsohn permutation (BTF) was leveraged in KLU (2004)
- In 2008, BTF structure was leveraged to create a new preconditioned iterative method
  - Great for CMOS memory circuits
  - Circuits with parasitics are more challenging
- In 2011, initial development of ShyLU, a "hybrid-hybrid" sparse linear solver package
  - Improve robustness





W. Bomhof and H.A. van der Vorst [NLAA, 2000]
A. Basermann, U. Jaekel, and K. Hachiya [SIAM LA 2003 proc.]

Sandia National Laboratories

"Hybrid-Hybrid" Linear Solvers

- ShyLU is a sparse linear solver framework, based on Schur complements (*S. Rajamanickam, E. Boman, M. Heroux*):
  - Incorporates both direct and iterative methods
  - Coarse-scale (multi-processor) and fine-scale (multi-threaded) parallelism
  - Can be a subdomain solver / preconditioner or stand-alone linear solver
- The Schur complement approach solves

$$Ax = b$$

by partitioning it into

$$A = \begin{bmatrix} D & C \\ R & G \end{bmatrix}, x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix},$$

where D and G are square, D is non-singular, and x and b are conformally partitioned

- The Schur complement is: $S = G - R * D^{-1}C.$

Sandia National Laboratories

“Hybrid-Hybrid” Linear Solvers

- Solving $Ax = b$ consists of three steps:

  1. Solve $Dz = b_1$.
  2. Solve $Sx_2 = b_2 - Rz$.
  3. Solve $Dx_1 = b_1 - Cx_2$.

- For Xyce, ShyLU is used as a stand-alone solver
  - Matrices partitioned using hypergraph partitioning (Zoltan)
    - Wide separator
  - D is solved exactly using KLU
  - S is solved iteratively via GMRES with S' as a preconditioner
    - S' generated through dropping
    - Maximum number of iterations = 30

ShyLU & Xyce Results

- This solution approach was necessary for efficient simulation of a Sandia-designed ASIC:
  - 1645693 total devices, N = 1944792
  - Single KLU solve takes ~ 40 sec.
  - ShyLU: 4 MPI procs -> number of rows in $S = 1854$

Nodes	Config. (MPI x threads)	ShyLU time (sec.)	Speedup (over KLU)	Total Sim. Time (sec.)	Speedup (over KLU)
1	4 x 2	61545	1.3x	66089	1.3x
1	2 x 4	61061	1.3x	68426	1.2x
2	8 x 2	23008	3.4x	27985	3.0x
2	4 x 4	35137	2.3x	40430	2.0x
3	12 x 2	17976	4.4x	22783	3.7x
3	6 x 4	26250	3.0x	33162	2.6x

Concluding Remarks

- Next-generation architectures demand new programming models for application codes
  - Re-evaluate simulation structure for intra-node parallelism
  - Third-party libraries can facilitate some of this transition
    - Trilinos: Epetra, Zoltan, ShyLU, …
- ShyLU can provide a flexible, robust solver framework for circuit simulation
  - What if diagonal blocks are singular? -> inner / outer schemes
  - Powernode parasitics can provide a need for narrow separators