SAND2019-3488C


# GOMACTech-2019

# Xyce: Open Source Simulation for Large-Scale Circuits

**3/28/2019**

**Eric Keiter**

**Sandia National Laboratories**

Sandia National Laboratories

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

# The Xyce™ Parallel Electronic Simulator Analog Circuit Simulator

- SPICE-Compatible syntax (Berkeley 3f5)
- Not “Fast SPICE”
- Two versions, **Serial** and…
- **Distributed Memory Parallel** (MPI-based)
- Unique solver algorithms
- Industry standard models
- Non-traditional models
  - Neuron/synapse
  - TCAD (PDE-based)

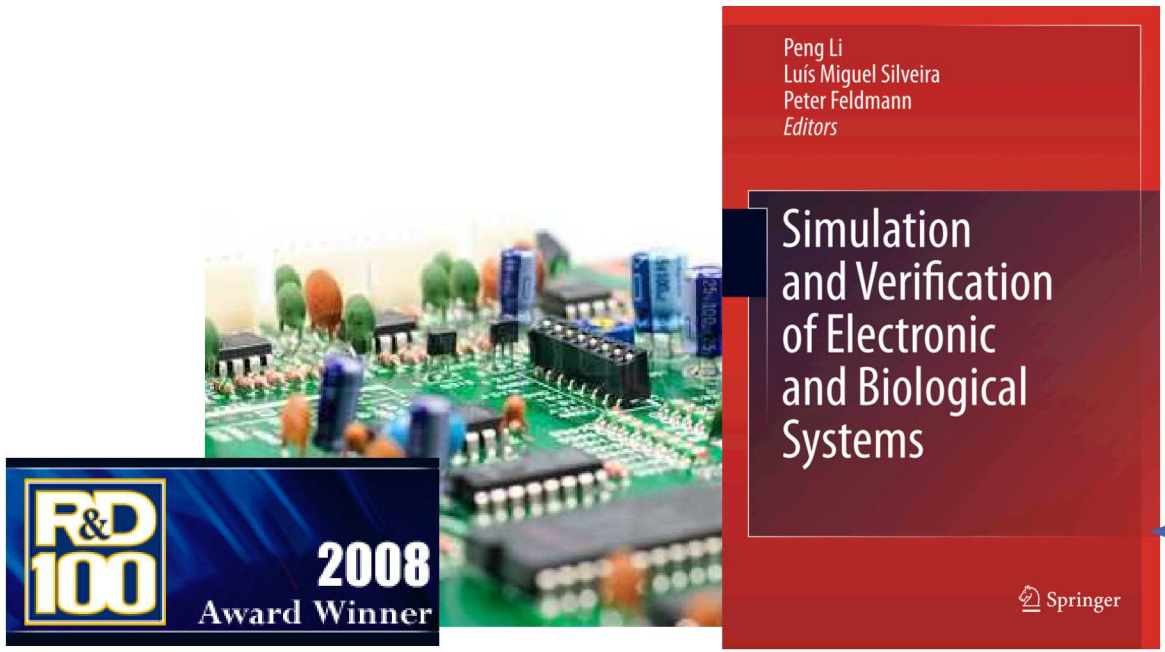
- http://xyce.sandia.gov

**Open Source, GPLv3**

- Since September of 2013 (Xyce 6.0)

Xyce Release 6.10

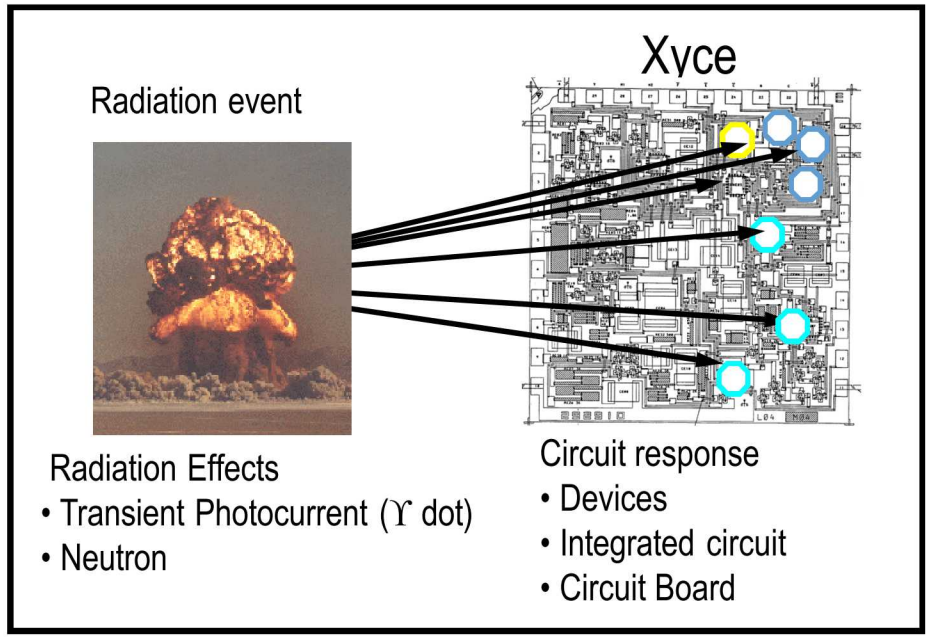
- November, 2018; 23rd major release
- >3,900 external downloads




Keiter, et al., “Parallel Transistor-Level Circuit Simulation”

# Original Xyce Motivation: Support Radiation Effects Prediction

- Sandia ensures electrical systems survivability in hostile environments
  - Comprehensive Test Ban Treaty (CTBT), 1993
  - Advanced Simulation & Computing (ASC), 1995
  - Qualification Alternatives to SPR (QASPR), 2005
- Requires electrical modeling:
  - Prompt radiation performance
  - Extended lifecycle (20 to 40 years)
- Unique Requirements ➡ Differentiating capabilities
  - Unique models: Radiation Effects
  - High fidelity: SPICE-level or higher
  - Large capacity: Massively-parallel
  - IP: Sandia owns it




✓ **Advanced mathematical algorithms**

✓ **Massively parallel applications**

# Why Open Source?

- First open source release, v6.0
- November 5, 2013.
- GPL license v3.0
- Source and binary downloads available
- xyce.sandia.gov

- Foster external collaboration
- Feedback from wider community
- Taxpayer funded, so encouraged  to open source.

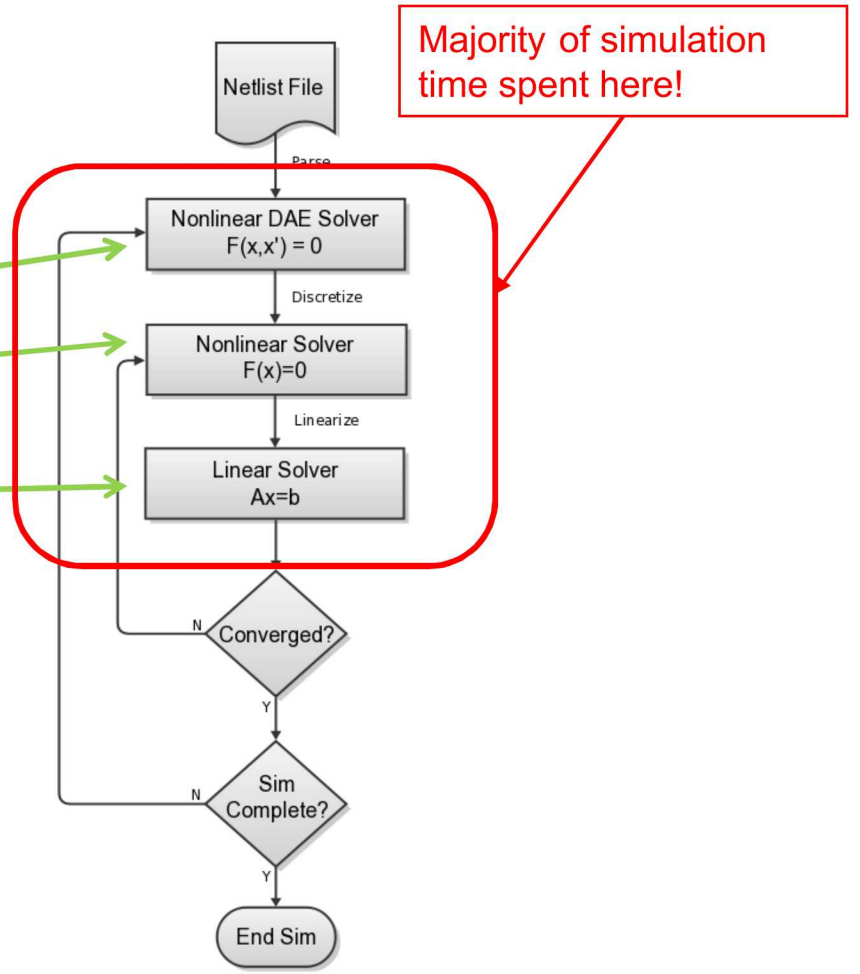
# Xyce Parallel Design

- Design Issues/Concerns:
  - Crucial to design parallel "from the ground up"
  - More flexibility if parallel partitioning is done on matrix level
    - But want to exploit circuit hierarchy if possible
  - Different parallel partitioning for different aspects of the problem
  - Sandia's expertise: large scale parallel iterative linear solvers
    - Direct methods important for circuits as well, if below a certain size
  - Scaling:
    - Distributed memory scales better than shared memory.
    - Iterative linear solvers scale better than direct.
  - Emergence of multicore technology.
    - Parallel computing no longer just supercomputers
    - Is MPI enough or do we incorporate options for TBB, CUDA, etc.?

# Circuit Simulation Flow

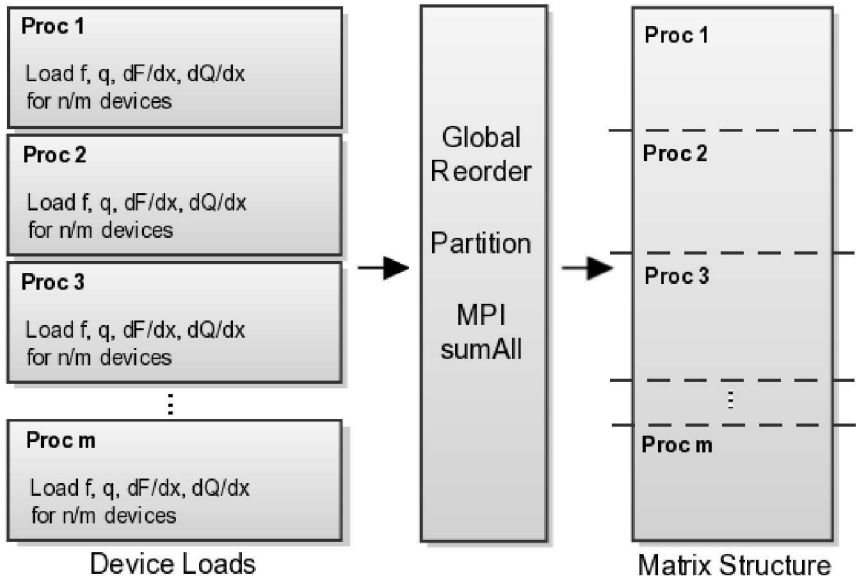
- Circuit simulators solve a system of nonlinear DAEs
  - How this is done depends on analysis type
  - Implicit integration methods
  - Newton's method
  - Sparse matrix techniques

- Transient simulation has several phases
  - Compute starting point (DCOP)
  - Start analysis (transient)
  - Sparse linear algebra / solvers
    - •Lynchpin of scalable performance



# 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



# Circuit Simulation Flow
## (Load Balancing)

**Dominating cost for small networks**

**Dominating cost for large networks**

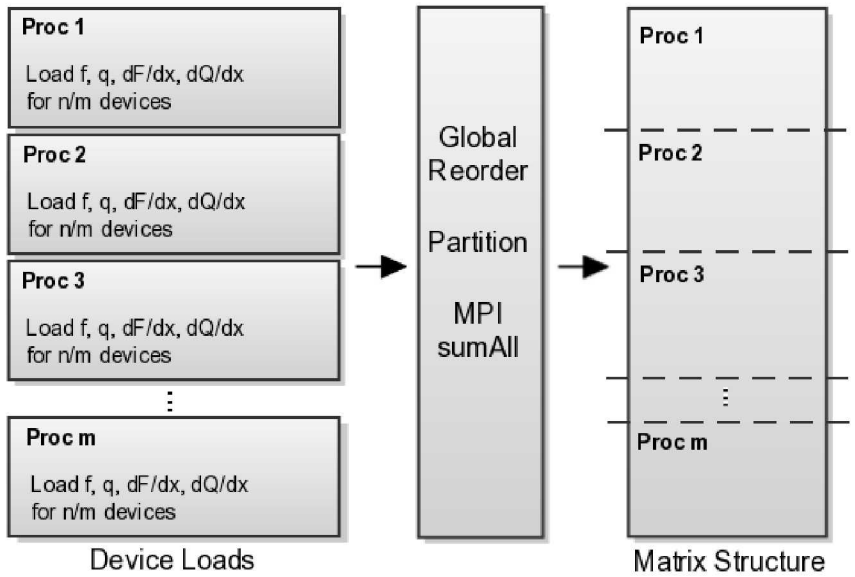
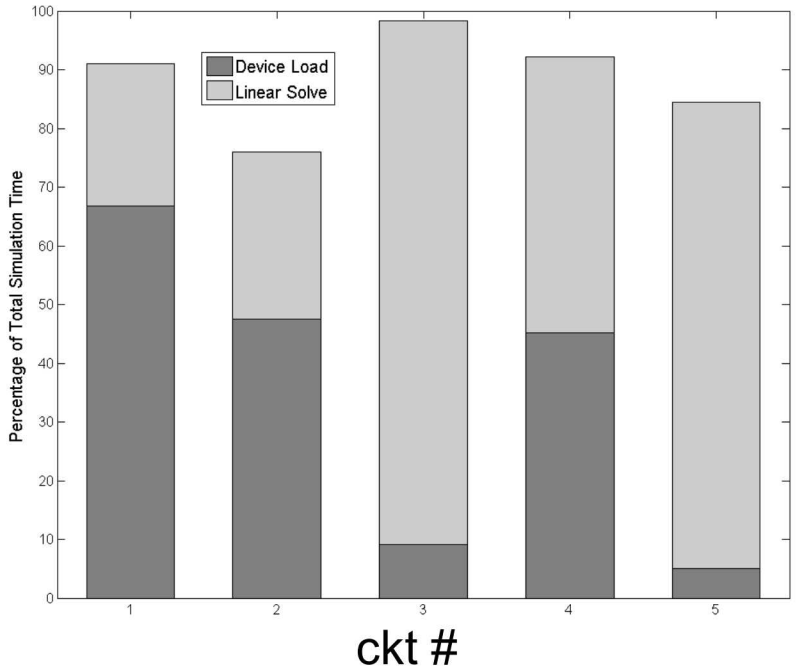



TABLE I
CIRCUITS: MATRIX SIZE(N), CAPACITORS(C), MOSFETS(M), RESISTORS(R), VOLTAGE SOURCES(V), DIODES (D).

| Circuit | N | C | M | R | V | D |
|---|---|---|---|---|---|---|
| ckt1 | 15622 | 7507 | 10173 | 11057 | 29 | 0 |
| ckt2 | 25187 | 0 | 71097 | 0 | 264 | 0 |
| ckt3 | 116247 | 52552 | 69085 | 76079 | 137 | 0 |
| ckt4 | 688838 | 93 | 222481 | 176 | 75 | 291761 |
| ckt5 | 1944792 | 400234 | 211486 | 795827 | 36100 | 199992 |

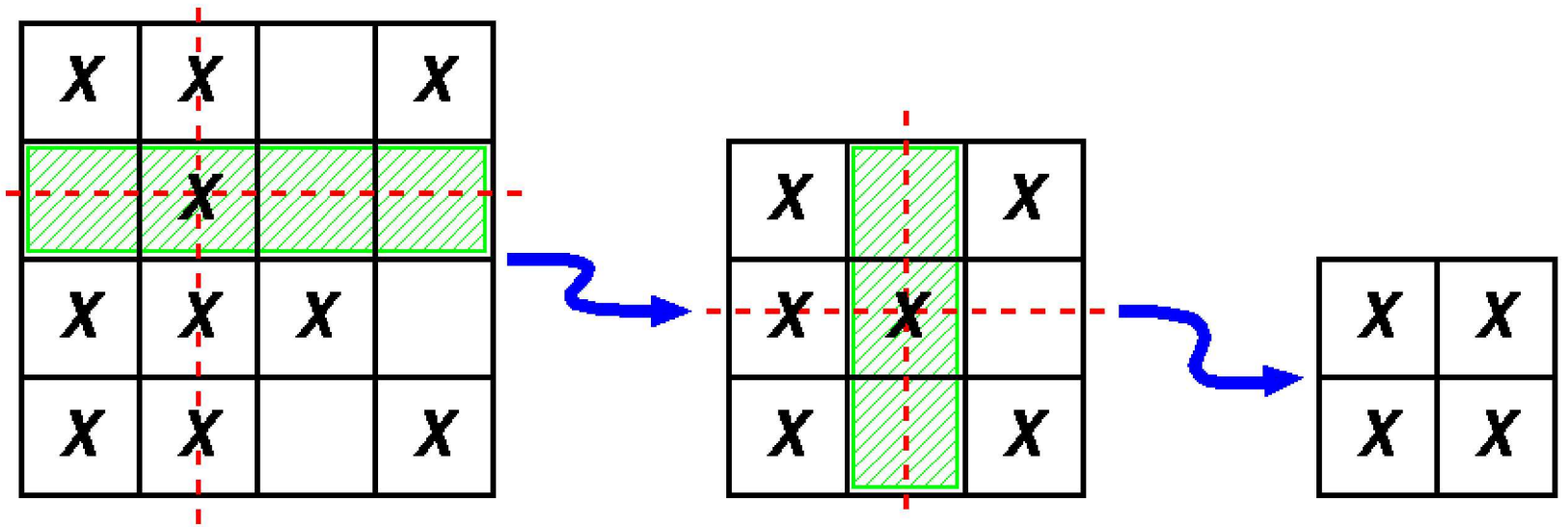


# Singleton Removal: Problem of dense rows and columns (1)

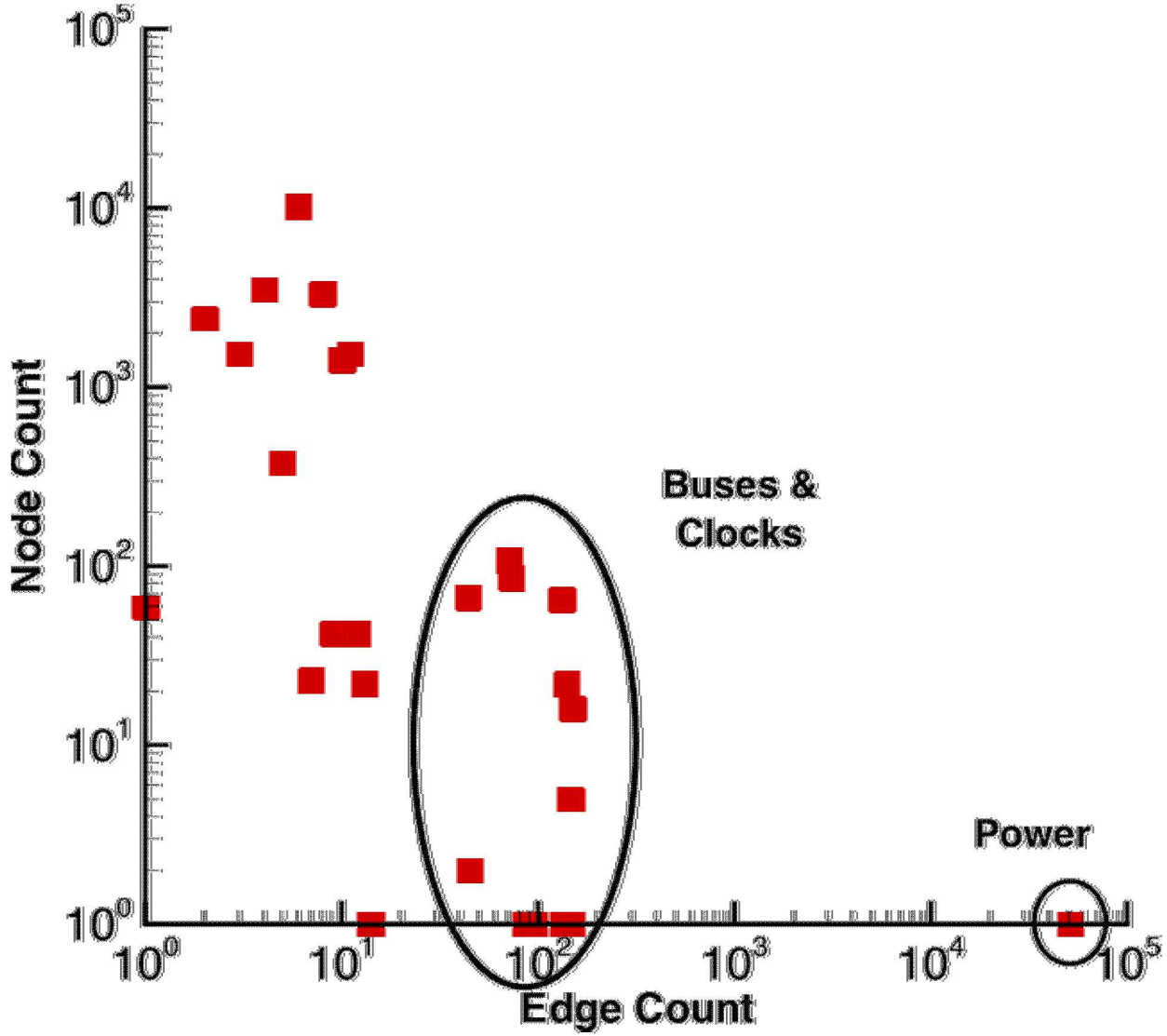
Very sparse linear systems but some dense rows and columns

Problematic for: reordering, repartitioning, load balance, condition

Solution: *Global node* removal

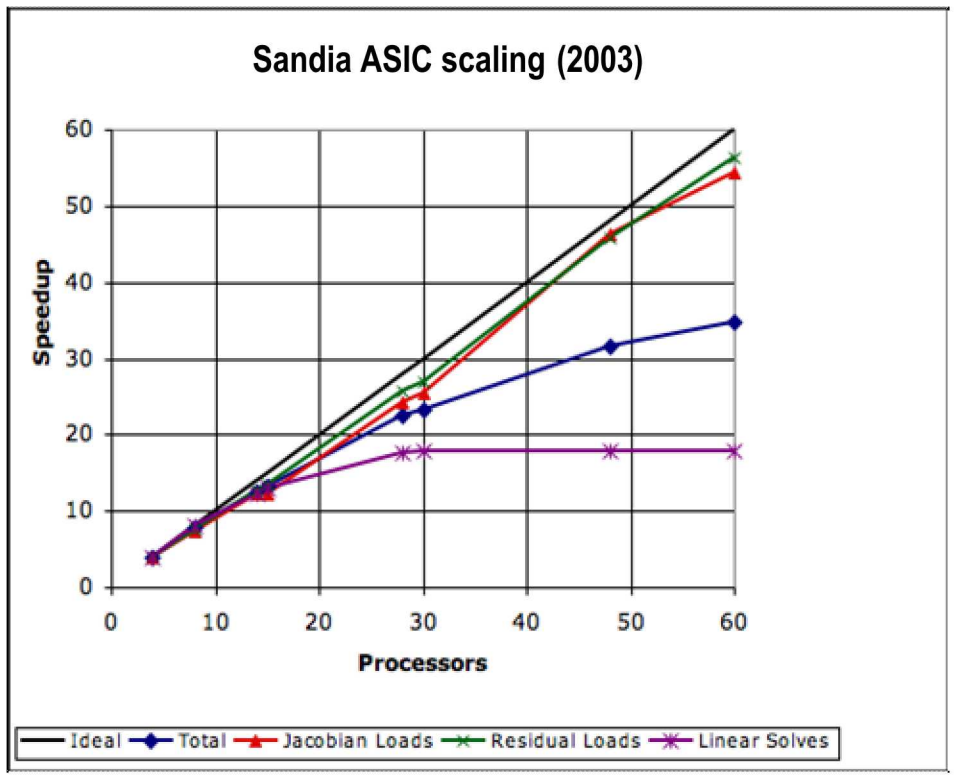


# Singleton Removal: Problem of dense rows and columns (2)



# Linear Solver Scaling / Robustness

- Initially (circa 1999), Xyce used available PDE-based preconditioning techniques
  - Incomplete LU factorization on subdomains
  - Limited scaling / robustness
  - Useful for simulating CMOS ASICs
    - *Homotopy techniques were necessary for the simulation to complete*

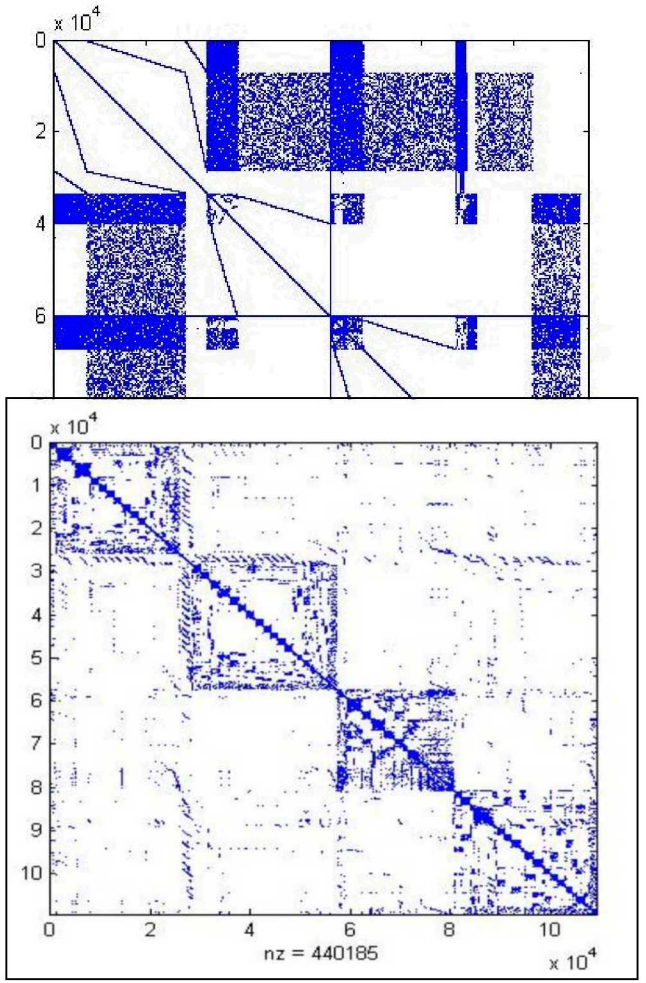



- For small scale circuits (like RHP), the Dulmage-Mendelsohn permutation (BTF) was leveraged to develop a direct solver (KLU, 2004)

# Linear Solver Scaling / Robustness

- In 2008, BTF structure was leveraged to create a new preconditioned iterative method
  - Great for CMOS memory circuits
  - Not effective on all circuits

| Circuit | Task | KLU (serial) | SLUD | DD | BTF | **Speedup** (KLU/BTF) |
|---|---|---|---|---|---|---|
| ckt1 | Setup | 2396 | $F_3$ | 207 | 199 | 12.0x |
| | Load | 2063 | $F_3$ | 194 | 180 | 11.4x |
| 16 procs | Solve | 1674 | $F_3$ | 3573 | 310 | 5.4x |
| | **Total** | 6308 | $F_3$ | 4001 | 717 | 8.8x |
| ckt7 | Setup | 57 | 21 | 21 | 22 | 2.6x |
| | Load | 801 | 219 | 218 | 221 | 3.6x |
| 4 procs | Solve | 346 | 318 | 280 | 67 | 5.2x |
| | **Total** | 1360 | 606 | 567 | 360 | 3.8x |




- Consider hybrid direct / iterative techniques
  - Improve robustness, how about scalability?

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

# CMOS ViArray Circuit

- Sandia CMOS ViArray ASIC
  - 549144 devices (N=434749)
  - Current preconditioned iterative methods fail
    - BTF: irreducible block size = *334767*

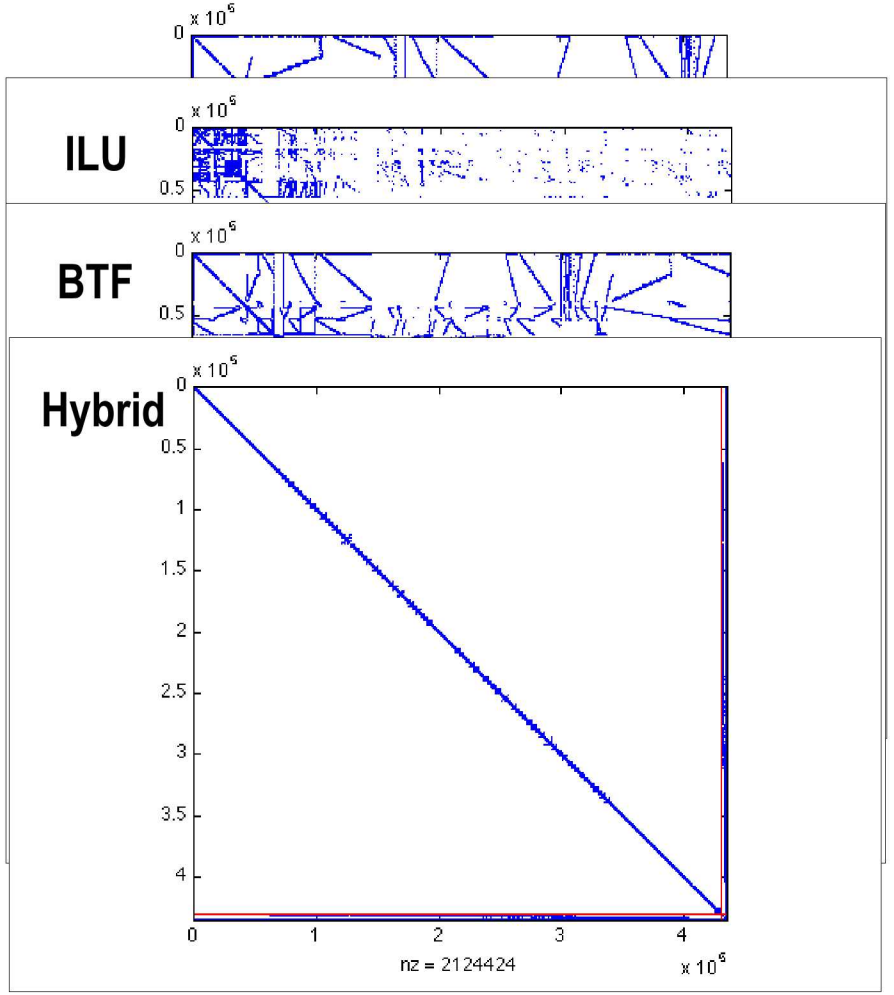
16 procs

BTF Fail

**Table 4: Simulation Times in Seconds on Sixteen Cores**

| Circuit | Task | KLU (serial) | SLUD | DD | BTF | Speedup (KLU/BTF) |
|---|---|---|---|---|---|---|
| ckt1 | Setup | 2396 | $F_3$ | 207 | 199 | 12.0x |
| | Load | 2063 | $F_3$ | 194 | 180 | 11.4x |
| | Solve | 1674 | $F_3$ | 3573 | 310 | 5.4x |
| | **Total** | 6308 | $F_3$ | 4001 | 717 | 8.8x |
| ckt2 | Setup | 2676 | $F_2$ | $F_2$ | $F_1$ | |
| | Load | 1247 | $F_2$ | $F_2$ | $F_1$ | |
| | Solve | 1273 | $F_2$ | $F_2$ | $F_1$ | |
| | **Total** | 5412 | $F_2$ | $F_2$ | $F_1$ | |

- BTF method not effective on ckt2
- Using Bomhof's hybrid partitioning:
  - *Direct : 429974 rows*
  - *Iterative : 4775 rows*



# Framework for Developing Robust "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
- This 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, $x$ and $b$ are conformally partitioned

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

# Flexible Solution Approach with ShyLU

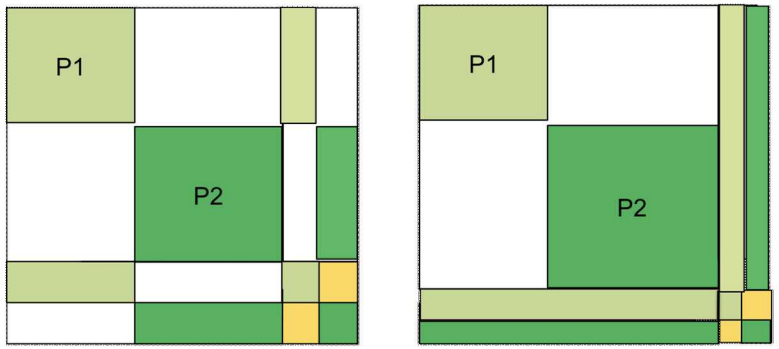
- Solving $Ax = b$ consists of three steps:

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

*D* solved exactly using KLU

*S* solved iteratively via preconditioned GMRES

- ShyLU is used as a stand-alone solver in Xyce
  - Matrices partitioned using hypergraph partitioning (Zoltan)
    - Wide separator – *S* can be computed locally
    - Narrow separator – *S* is smaller, but requires communication
  - Preconditioner, *S'*, generated by dropping small entries in *S*



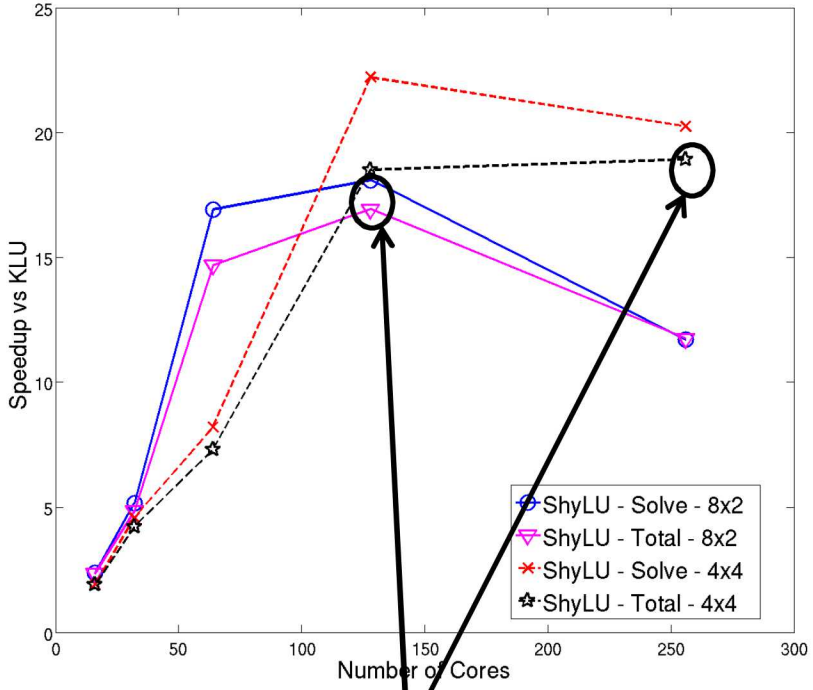
# Hybrid Solver (ShyLU) Achieves 19x Speedup for Shasta 2x2 Simulation

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

Shasta 2x2 ASIC: 1.6M total devices, ~ 2M unknowns:

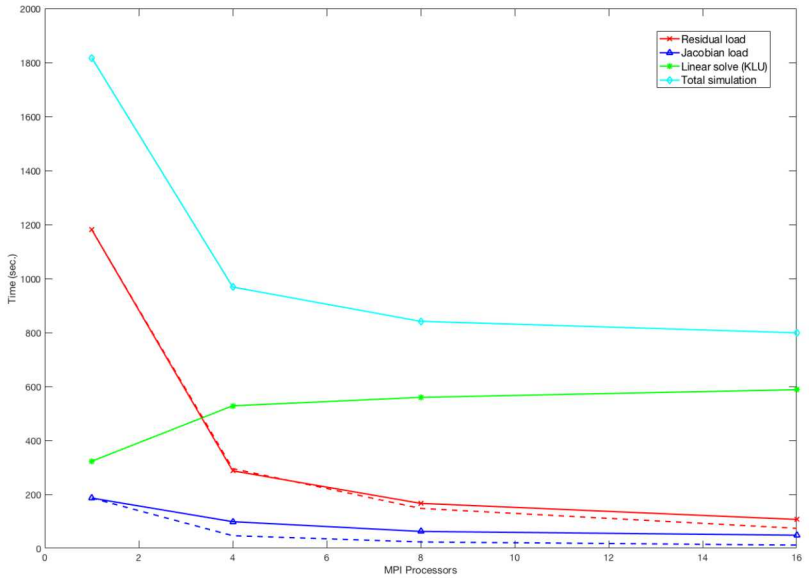
- Xyce w/ KLU solver takes **~ 2 weeks**
- **Xyce** w/ ShyLU solver takes **~ 1 day**
- ShyLU: Optimal # partitions = 64; number of rows in *S* = 1854 (4 MPI procs)



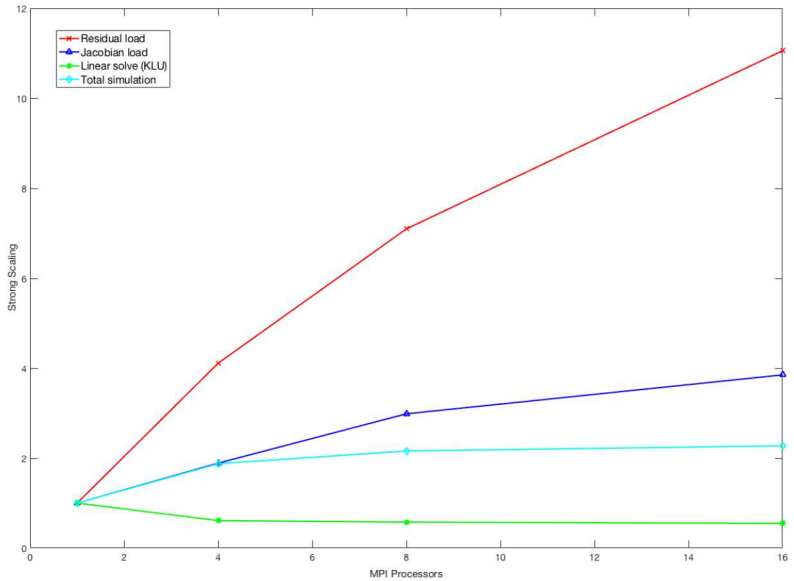

Strong scaling of Xyce's simulation time and ShyLU linear solve time for different configurations of MPI Tasks X Threads per node on TLCC

# 100k device memory circuit scaling results on Chama Circuit produced by Yale's memory compiler

## Runtime



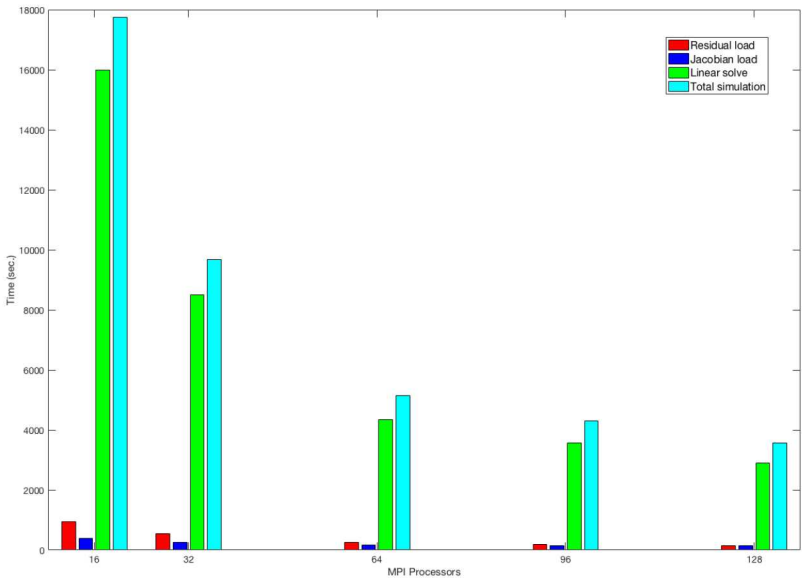

## Strong scaling



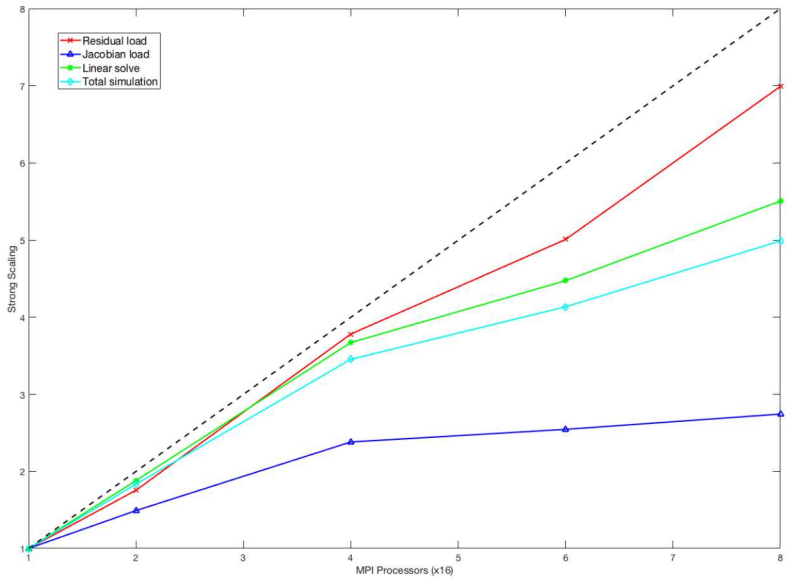

- The 100k scaling results are for parallel device evaluation / serial solve (KLU)
  - For this size of circuit (and smaller) the KLU serial direct solver beats preconditioned iterative solver.
- This was performed on Chama: Xtreme-X GreenBlade GB512X, Xeon E5-2670 8C 2.600GHz, Infiniband QDR

# 1M device memory circuit scaling results on Chama
# Circuit produced by Yale's memory compiler

## Runtime




## Strong scaling




- The 1M scaling results are for parallel device evaluation/parallel preconditioned iterative solve (block Jacobi)
- This was performed on Chama: Xtreme-X GreenBlade GB512X, Xeon E5-2670 8C 2.600GHz, Infiniband QDR

# Xyce Team Acknowledgements


- Eric R. Keiter
- Thomas V. Russo
- Richard L. Schiek
- Heidi K. Thornquist
- Ting Mei
- Jason C. Verley
- Peter E. Sholander
- Karthik V. Aadithya
- …and many others


Contact:

`http://xyce.sandia.gov`
`xyce@sandia.gov`

Google Group Forum:

`https://groups.google.com/forum/#!forum/xyce-users`

# Obtaining Xyce

- We don't yet have a public git repository. (this will happen soon)
- Go to our website: Click on the download button, fill out form
- At download page, many resources are available.
  - **Binary executables** for Windows, OSX and Red Hat Enterprise Linux
  - **Xyce Source code** and **build instructions**
    - If you do this, follow instructions carefully.
    - You must build the Trilinos library with the EXACT options we specify.
    - https://trilinos.org/
  - **Regression test suite** (several thousand tests)
  - **Documentation**
    - Users guide
    - Reference guide
    - Mathematical Formulation
    - Release notes
    - FAQ
    - Xyce/ADMS Users guide
    - Hpice compatibility App note

http://xyce.sandia.gov

# Questions?