SAND2013-10491P

Xyce Parallel Circuit Simulator

Eric Keiter
MOS-AK Workshop
Washington, DC
Dec. 11, 2013

Outline

Background/Motivation✔

Xyce overview:

- Xyce v6.0 open-source release✔
- Parallel design ✔

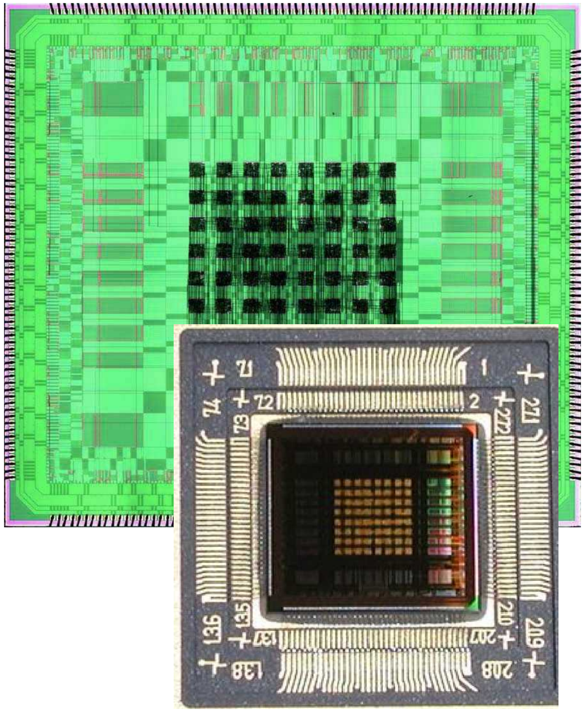
Compact models:

- ADMS model compiler ✔
- ModSpec integration✔
- Xyce supports most SPICE models (BSIM, EKV, etc)
- Neutron effects
- Photocurrent/Photoconductivity

✔ Covered in this talk

Xyce™ Parallel Circuit Simulator

- Xyce: Massively Parallel circuit simulator:
  - SPICE-Compatible
  - Industry standard models (BSIM, PSP, EKV, VBIC, FBH, etc)
  - Distributed Memory Parallel (MPI-based)
  - Unique solver algorithms
- Analysis types
  - DC, TRAN, AC
  - Harmonic Balance (HB)
  - Multi-time PDE (MPDE)
  - Model order reduction (MOR)
  - Direct and Adjoint sensitivity analysis
- Sandia-specific models
  - Prompt Photocurrent
  - Prompt Neutron
  - Thermal
- Xyce Release 6.0
  - Open Source!
  - GPL v3 license

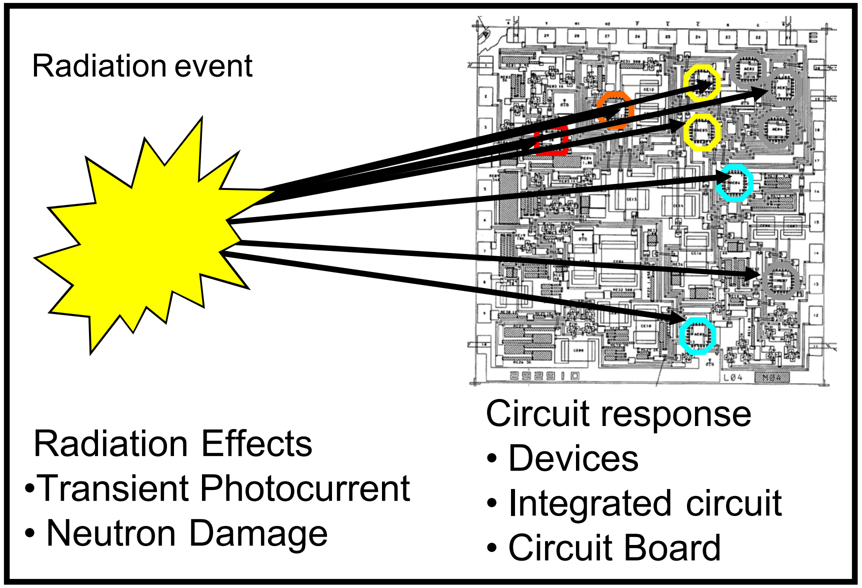


http://xyce.sandia.gov

Project Motivation: Why Xyce?

- Comprehensive Test Ban Treaty (CTBT), 1993
- Advanced Simulation & Computing (ASC), 1995
- Qualification Alternatives to SPR (QASPR), 2005
  - Offset lack of NW testing
  - Help qualify NW systems

- Unique Requirements ➡ Differentiating capabilities
  - Unique models: Radiation Effects
  - High fidelity: SPICE-level or higher
  - Large capacity: Massively-parallel
  - IP: Sandia owns it



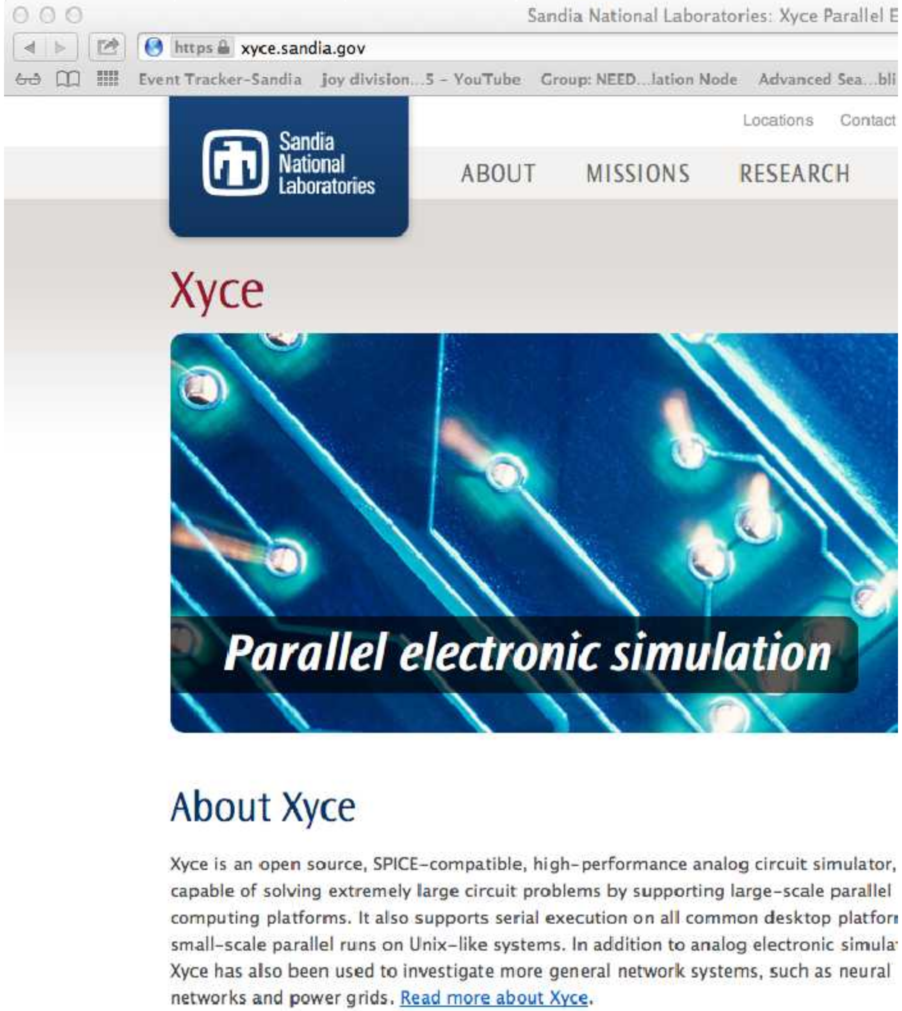
Radiation event

Radiation Effects
- Transient Photocurrent
- Neutron Damage

Circuit response
- Devices
- Integrated circuit
- Circuit Board

Xyce Open Source release

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


Sandia National Laboratories: Xyce Parallel E
https xyce.sandia.gov
Event Tracker-Sandia
Joy division...5 - YouTube
Group: NEED...lation Node
Advanced Sea...bli
Sandia National Laboratories
Locations
Contact
ABOUT
MISSIONS
RESEARCH

Xyce

Parallel electronic simulation

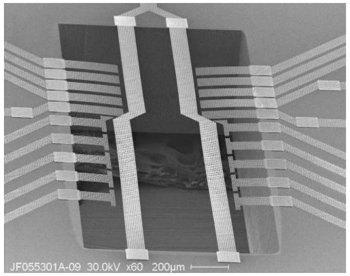
About Xyce

Xyce is an open source, SPICE-compatible, high-performance analog circuit simulator, capable of solving extremely large circuit problems by supporting large-scale parallel computing platforms. It also supports serial execution on all common desktop platfor small-scale parallel runs on Unix-like systems. In addition to analog electronic simula Xyce has also been used to investigate more general network systems, such as neural networks and power grids. Read more about Xyce.

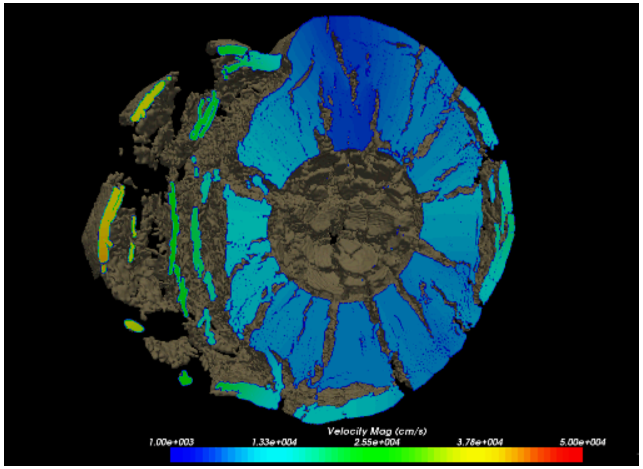
Xyce Features

- Designed for massively parallel systems.
  - Distributed parsing of very large netlists.
  - Distribution of problem across many processors.
  - Scalable performance.
- Advanced direct and iterative linear solvers.
- Advanced DCOP solution algorithms.
- Multilevel solve for power-node parasitics.
- Improved time integration methods.
- Integration with Dakota optimization tools.
- Physics-based radiation effects models.
- Extensible device package.

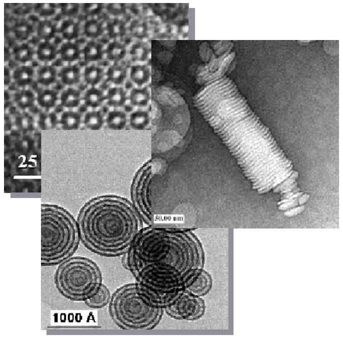
SNL has six core technical capabilities



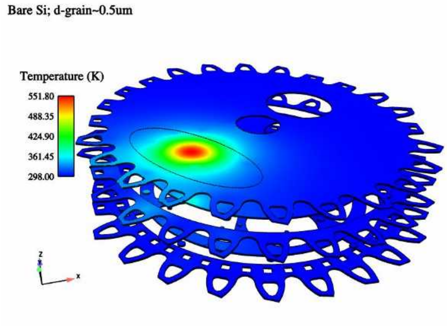
Microelectronics
and Photonics



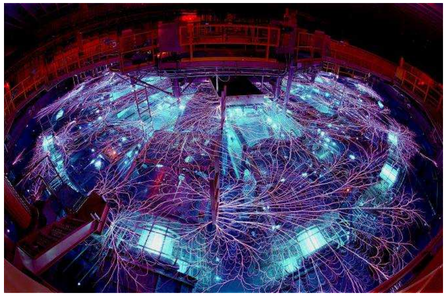
Computational &
Informational Sciences



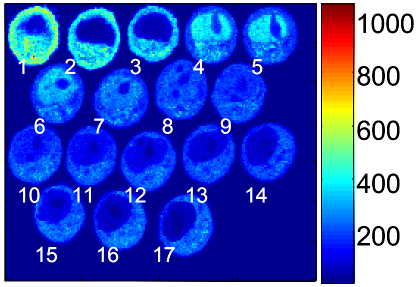
Materials Science &
Technology



Engineering Sciences



Pulsed Power

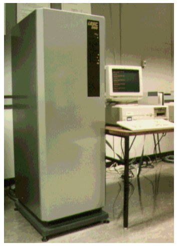


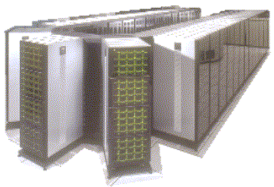
Bioscience

CIS has a rich history in the development and maturation of high performance computing hardware and software technology

CM-2

nCUBE-2


iPSC-860


Paragon

ASCI Red

Cplant

Red Storm

1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007

Gordon Bell Prize

R&D 100
Parallel Software

Karp Challenge

Patent
Parallel Software

R&D 100
Meshing

Patent
Meshing

Gordon Bell Prize

R&D 100
Signal Processing

World Record
143 GFlops

World Record
281 GFlops

R&D 100
Dense Solvers

R&D 100
Storage

Patent
Paving

SC96 Gold Medal
Networking

R&D 100
Aztec

World Record
Teraflops

Gordon Bell Prize

Patent
Decomposition

R&D 100
Salvo

Patent
Data Mining

Mannheim
SuParCup

R&D 100
Trilinos

R&D 100
Allocator

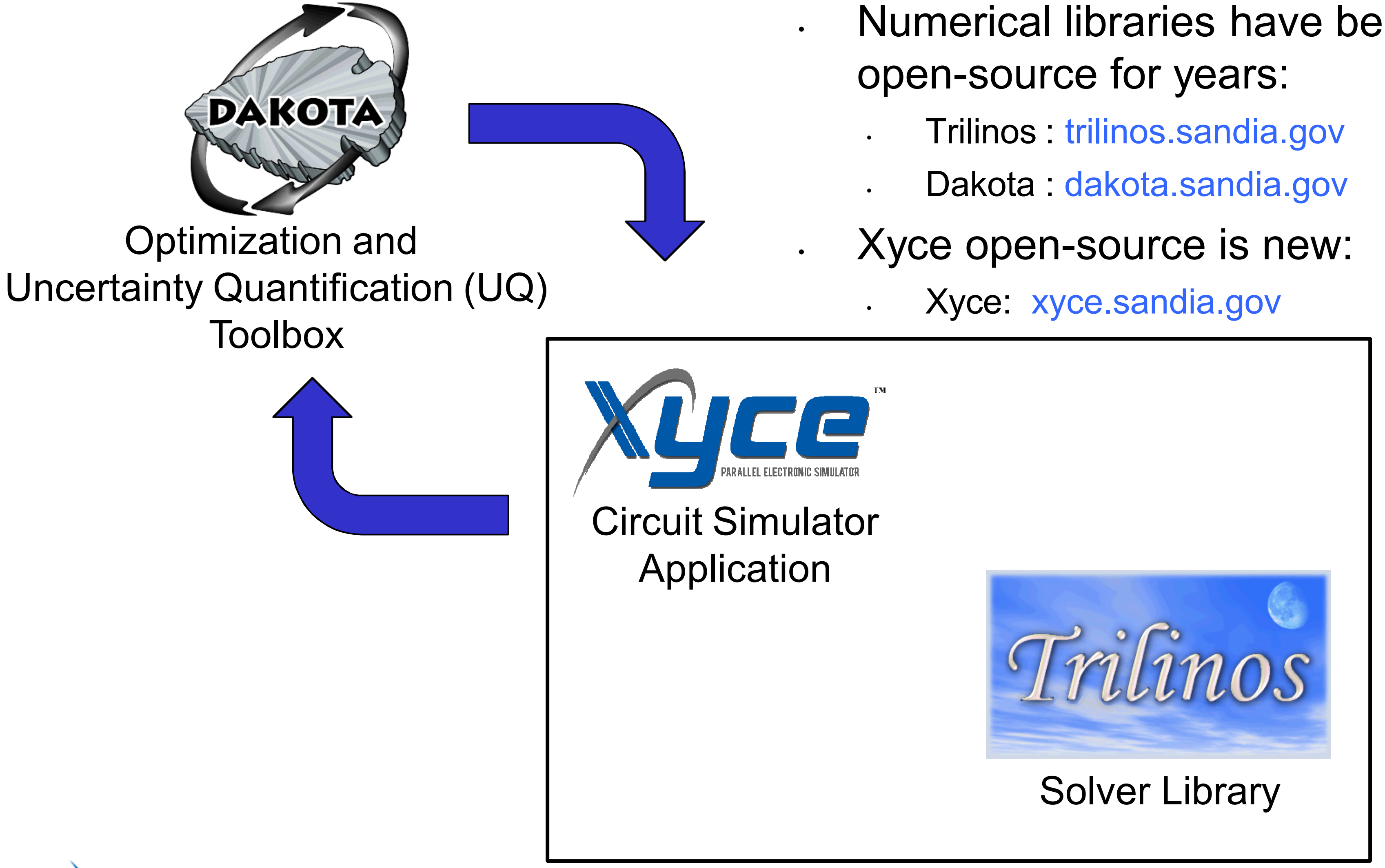
R&D 100
Xyce

R&D 100
3D-Touch

Fernbach
Award

Open-Source Capabilities

- Numerical libraries have been open-source for years:
  - Trilinos : trilinos.sandia.gov
  - Dakota : dakota.sandia.gov
- Xyce open-source is new:
  - Xyce: xyce.sandia.gov



Optimization and Uncertainty Quantification (UQ) Toolbox

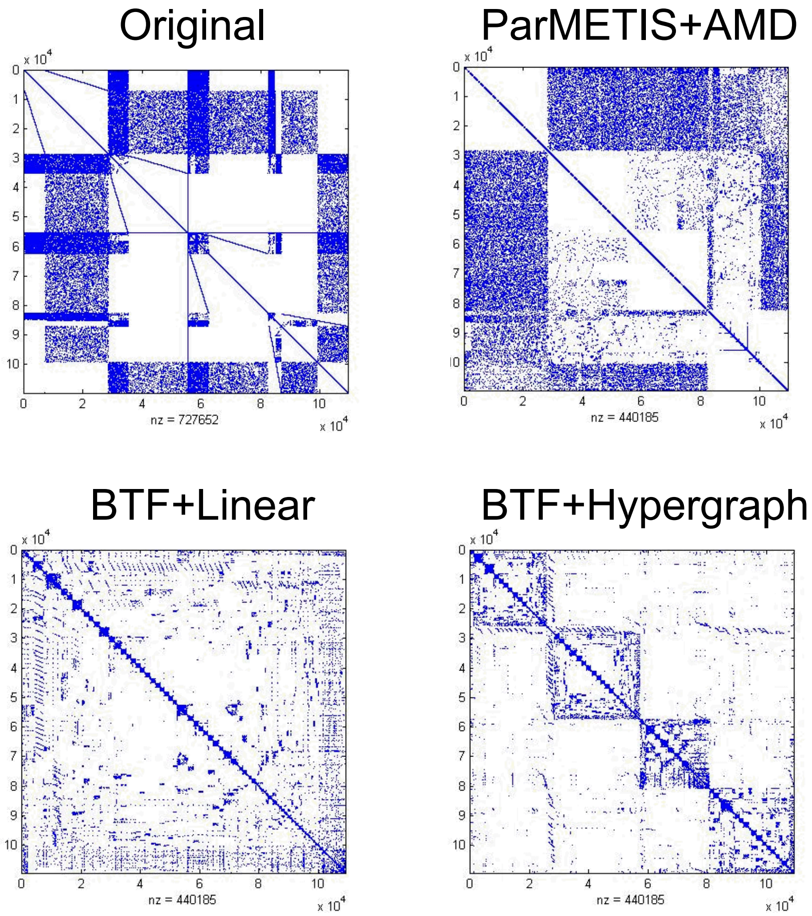
Circuit Simulator Application

Solver Library

Xyce Parallel Design

- Design Issues/Concerns:
  - Crucial to design parallel “from the ground up”
  - Capacity or Capability machines?
  - More flexibility if parallel partitioning is done on matrix level.
  - Sandia’s expertise: large scale parallel iterative solvers..
  - Scaling:
    - Distributed memory scales better than shared memory.
    - Iterative 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.?

Advanced Parallel Linear Solvers


Original

ParMETIS+AMD

BTF+Linear

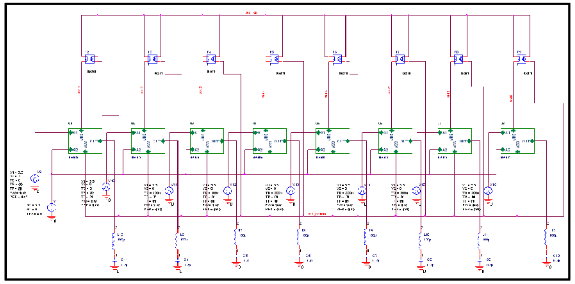
BTF+Hypergraph

100K Transistor IC Problem

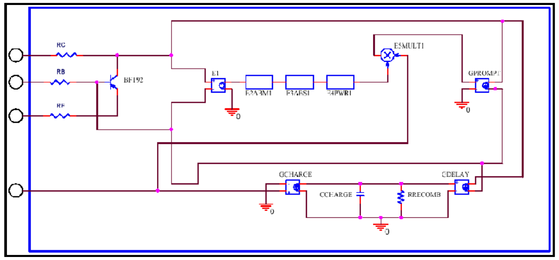
- Large circuits lead to large, sparse matrices with complex structure.
- Circuit problems produce matrices completely unlike those in other disciplines.
- Efficient solution of these linear systems requires new strategies for reordering, partitioning, and preconditioning.

Transforming Design Capabilities

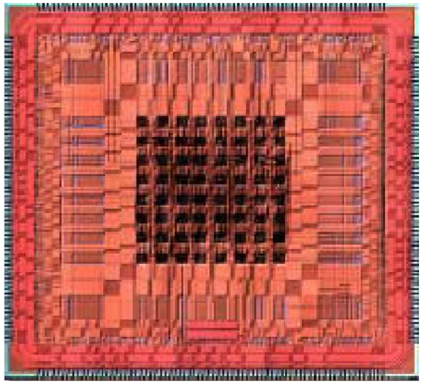
then ~ 2001
PSpice circuit model:
~300 devices



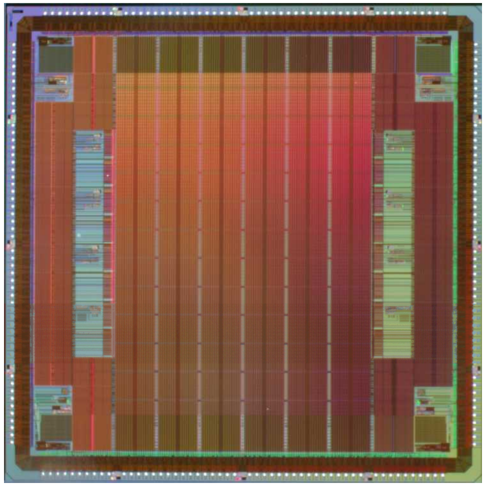
rad. model:
empirical/behavioral



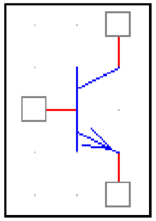
2006
Xyce circuit model:
300K+ devices



201
2
Xyce circuit model:
40M+ devices



rad. model:
physics-based/built-in



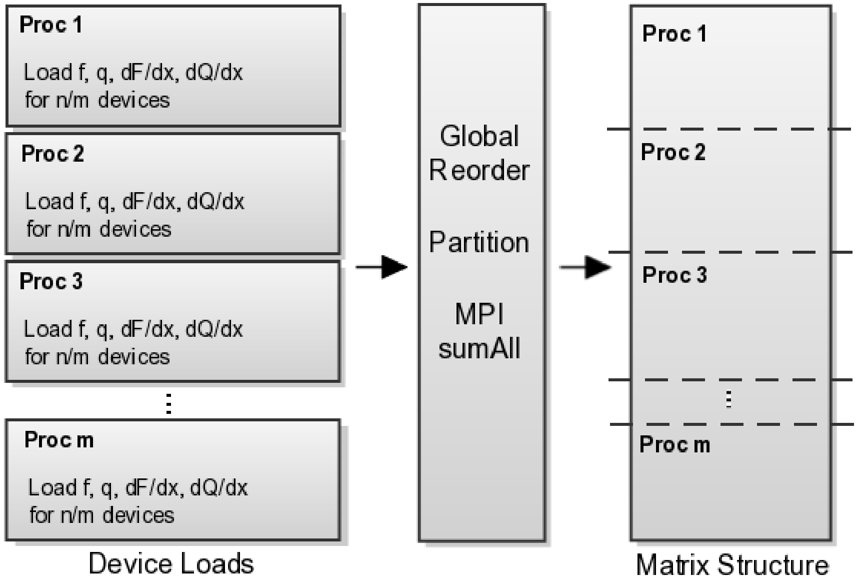
Eric Keiter, Sandia National Laboratories
2013 MOS-AK Workshop, Washington DC

Balancing Multiple Solver Objectives

- 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
  - Ghost node distribution can be irregular

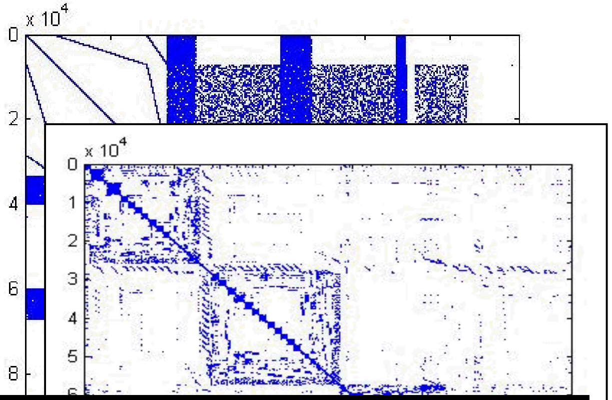
- Matrix Structure
  - Use graph structure to determine best preconditioners / solvers



Proc 1
Load f, q, dF/dx, dQ/dx for n/m devices
Proc 2
Load f, q, dF/dx, dQ/dx for n/m devices
Proc 3
Load f, q, dF/dx, dQ/dx for n/m devices
Proc m
Load f, q, dF/dx, dQ/dx for n/m devices
Device Loads
Global Reorder
Partition
MPI sumAll
Proc 1
Proc 2
Proc 3
Proc m
Matrix Structure

Xyce History of Linear Solvers

- 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 2010, initial development of ShyLU, a “hybrid-hybrid” sparse linear solver package
  - Improve robustness



Preconditioning Method	Residual	GMRES Iters	Solver Time (s.)
Local AMD ILUT ParMETIS	3.43e-01 (FAIL)	500	302.573
BTF Block Jacobi Hypergraph	3.47e-10	3	0.139

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

***26x* speedup on *16* cores**

Hybrid-Hybrid solver result: 19x Speedup for Challenging IC

Necessary for efficient simulation of a primary logic component for W88-Alt AF&F:

- 1.6M total devices, ~2M unknowns
- Xyce w/ KLU solver takes ~ **2 weeks**, w/ ShyLU solver takes ~ **1 day**
- ShyLU: Optimal # partitions = 64; number of rows in *S* = 1854 (4 MPI procs)

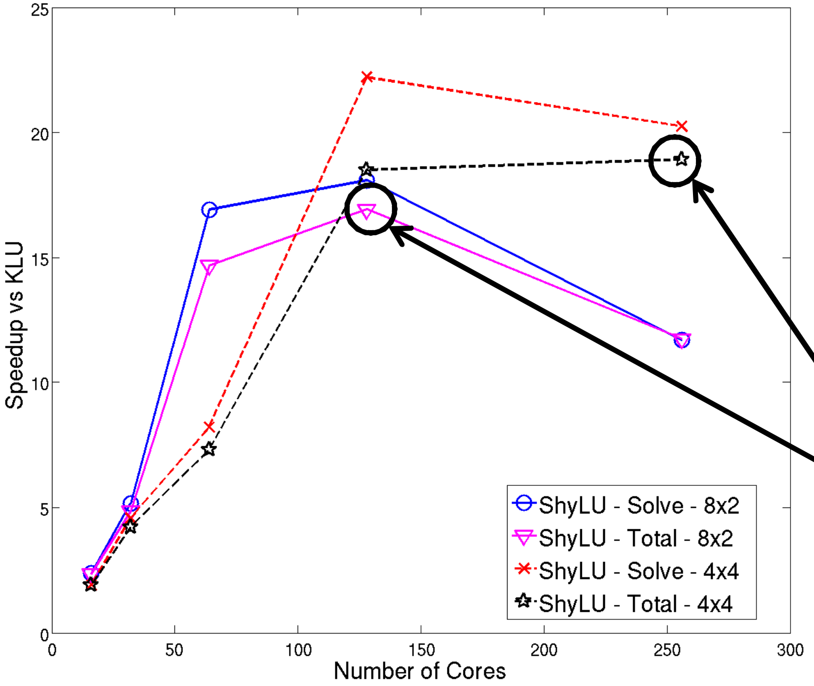


TABLE III
COMPARISON OF TOTAL LINEAR SOLVE TIME (SEC.) OF VARIOUS SPARSE DIRECT SOLVERS FOR OUR TEST CIRCUITS; (-) INDICATES SIMULATION FAILED TO COMPLETE.

	ckt1	ckt2	ckt3	ckt4	ckt5
KLU	**80.8**	162.2	9381.3	7060.8	**14222.7**
PARDISO (16)	128.6	**105.3**	**715.0**	**6690.5**	-
SuperLU	-	10294.1	-	-	72176.8
SuperLU_Dist (16)	-	-	-	-	-

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

ADMS-Xyce

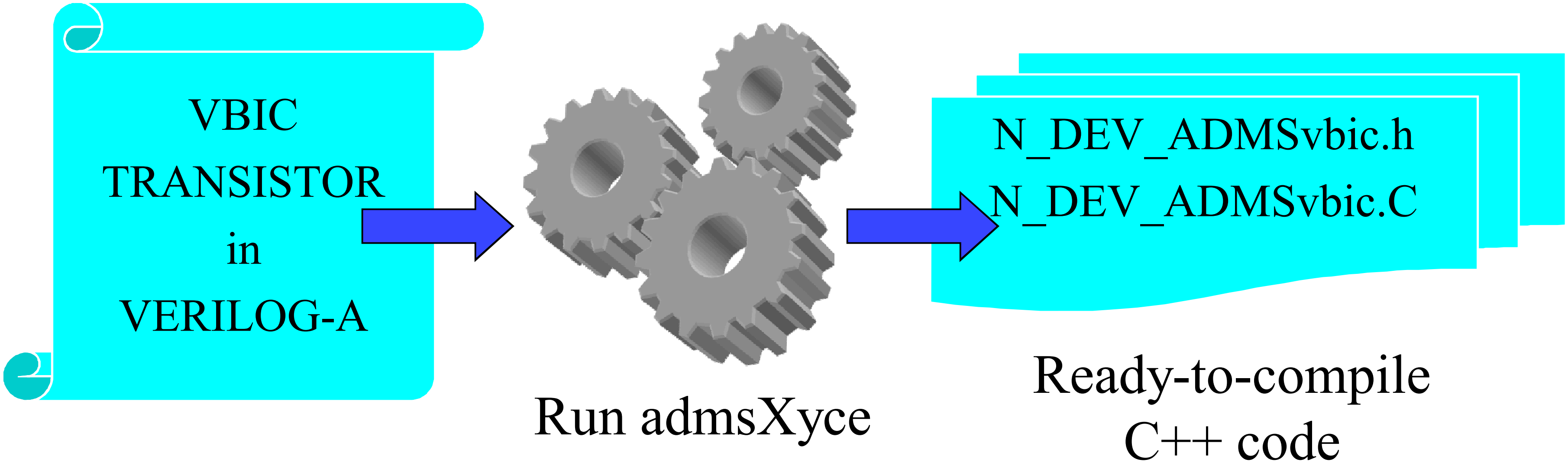
Verilog-A interface, via ADMS model compiler

- VBIC, Mextram, EKV, HiCUM, etc.

Verilog-A: industry standard format for new models

ADMS translates Verilog-A to compilable C/C++ code;

API automatically handles data structures, matrices, tedious details.



VBIC
TRANSISTOR
in
VERILOG-A

Run admsXyce

N_DEV_ADMSvbic.h
N_DEV_ADMSvbic.C

Ready-to-compile
C++ code

ADMS-Xyce

ADMS-converted models in Xyce:

- VBIC (the first model we did, only constant-phase so far)
- EKV (can't distribute thanks to NDA, so it's not in 6.0 open-source)
- PSP (version 103)
- BSIM-CMG 107 (only one of the many variants)
- FBH HBT_X version 2.1
- FBH HBT_X version 2.3 (we were able to process it into Xyce, but dropped this one after discussions with Matthias Rudolph (convergence problems))
- MEXTRAM (we don't support mextram, because its convergence issues require a voltage limiting approach not yet supported by our back-end, but the MEXTRAM verilog can be (and has been) processed by our ADMS back-end).

ADMS-Xyce

What is unique about Xyce's use of ADMS?

- Sacado AD library: By using Sacado, we are able to do away with an enormous amount of "admst" code for computing derivatives (compare with ng-spice's or qucs ADMS back-ends). It is difficult to overstate how much back-end work this saves.

- Have added "$limit" support (simple mod to ADMS required to let "$limit" be recognized as a legal function, but otherwise all the work is in the back-end)

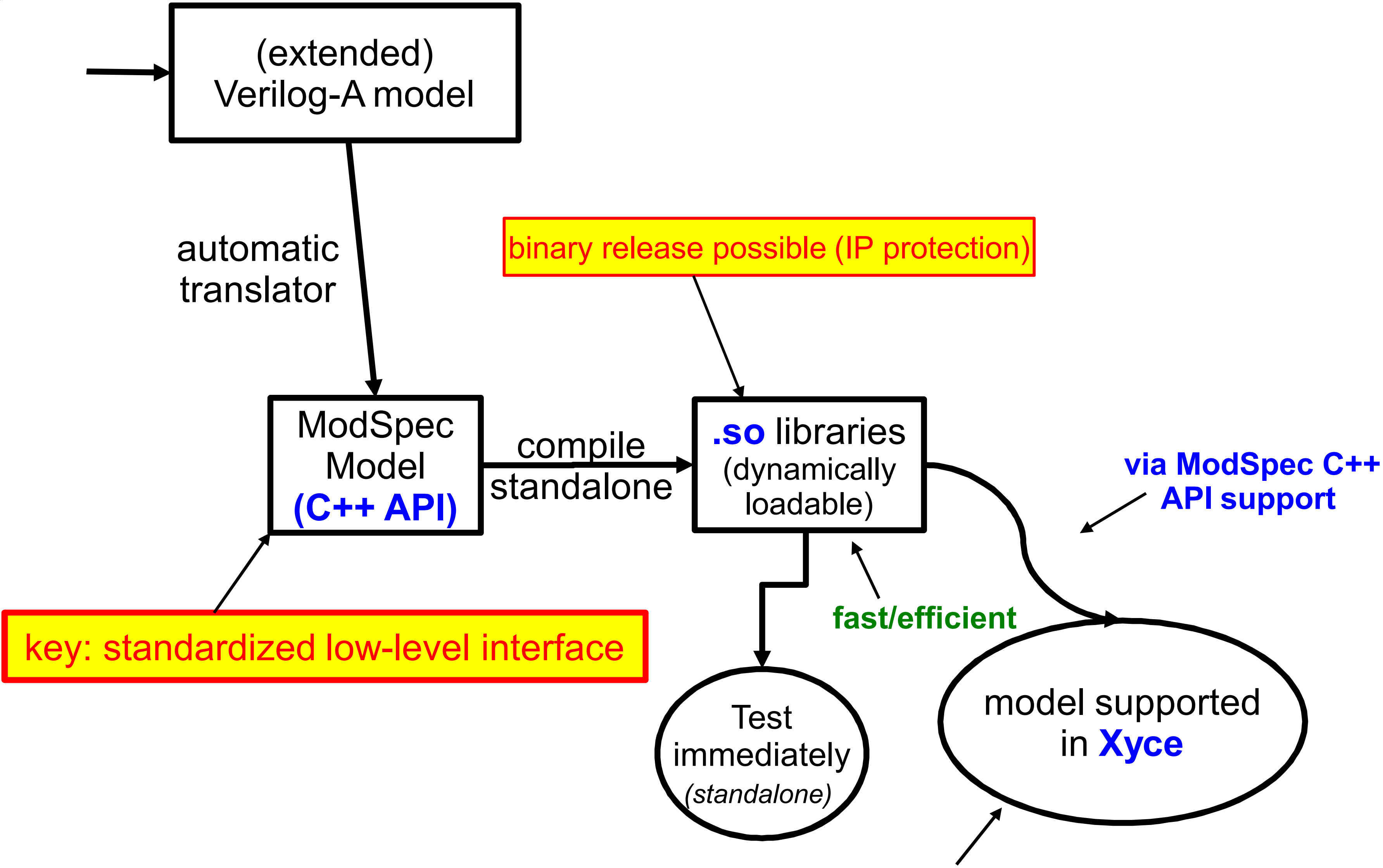
- Had to add some "attributes" for parameters and modules to provide certain metadata (descriptions, units, "model" vs. "instance", level number, whether it's a Q, M, Y device, etc.).

ADMS-Xyce

What challenges were there?

- ADMS back-ends are written in ADMST, a sort of XSLT superset.
- Xyce's linear system differs from SPICE so generating code for voltage limiting ($limit) is not as simple as generating a function call.
- ADMS has NO support for current branches. It generates datastructures representing the Jacobian resulting when all variables are nodal voltages, but you must do all the work yourself if you throw a branch current in the mix.
- Node collapse: since this has to happen at the time the device is instantiated (rather than the time its equations are evaluated), it is necessary to generate (separately from the rest of the evaluation) the parts of code needed to compute all the variables required to determine whether or not to collapse.

Using ModSpec in Xyce



(extended) Verilog-A model

automatic translator

ModSpec Model (C++ API)

compile standalone

.so libraries (dynamically loadable)

binary release possible (IP protection)

key: standardized low-level interface

via ModSpec C++ API support

fast/efficient

Test immediately *(standalone)*

model supported in Xyce

speed near native/hand-coded implementation (compiled C++ code)

Acknowledgements

Xyce team

xyce.sandia.gov

- Scott Hutchinson
- Tom Russo
- Heidi Thornquist
- Jason Verley
- Rich Schiek
- Ting Mei
- Dave Baur
- Sivasankaran Rajamanickam

Trilinos team

trilinos.sandia.gov

Dakota team

dakota.sandia.gov

Publications / Presentations

Publications

- *“Electrical Modeling and Simulation for Stockpile Stewardship”,* ACM XRDS, 2013
- *“ShyLU: A Hybrid-Hybrid Solver for Multicore Platforms”*, IPDPS 2012
- *“Parallel Transistor-Level Circuit Simulation”,* Simulation and Verification of Electronic and Biological Systems, Springer, 2011
- *“A Parallel Preconditioning Strategy for Efficient Transistor-Level Circuit Simulation”*, ICCAD 2009

Presentations

- *“Sparse Matrix Techniques for Next-Generation Parallel Transistor-Level Circuit Simulation”*

  Heidi K. Thornquist, Parallel Matrix Algorithms and Applications 2012
- *“Partitioning for Hybrid Solvers: ShyLU and HIPS”*

  Erik G. Boman, Siva Rajamanickam, and Jeremie Gaidamour, Copper Mtn. 2012
- *“Efficient Preconditioners for Large-Scale Parallel Circuit Simulation”*

  Heidi K. Thornquist, SIAM Computational Science & Engineering 2011
- *“Advances in Parallel Transistor-Level Circuit Simulation”*

  Heidi K. Thornquist, Scientific Computing in Electrical Engineering 2010
- *“Large Scale Parallel Circuit Simulation”*

  Heidi Thornquist and Eric Keiter, Circuit and Multi-Domain Simulation Workshop, ICCAD 2009

XRDS

Scientific Computing

Backup slides

Building on SPICE

Xyce starts with basic SPICE compatibility:

- PSPICE- and SPICE3F5-compatible netlist format (some HSPICE compatibility)
- SPICE3F5 standard models + BSIM3, BSIM4, B3SOI
- Standard time integration techniques (Trapezoid, Gear)
- Standard nonlinear solver techniques
- Direct linear solvers

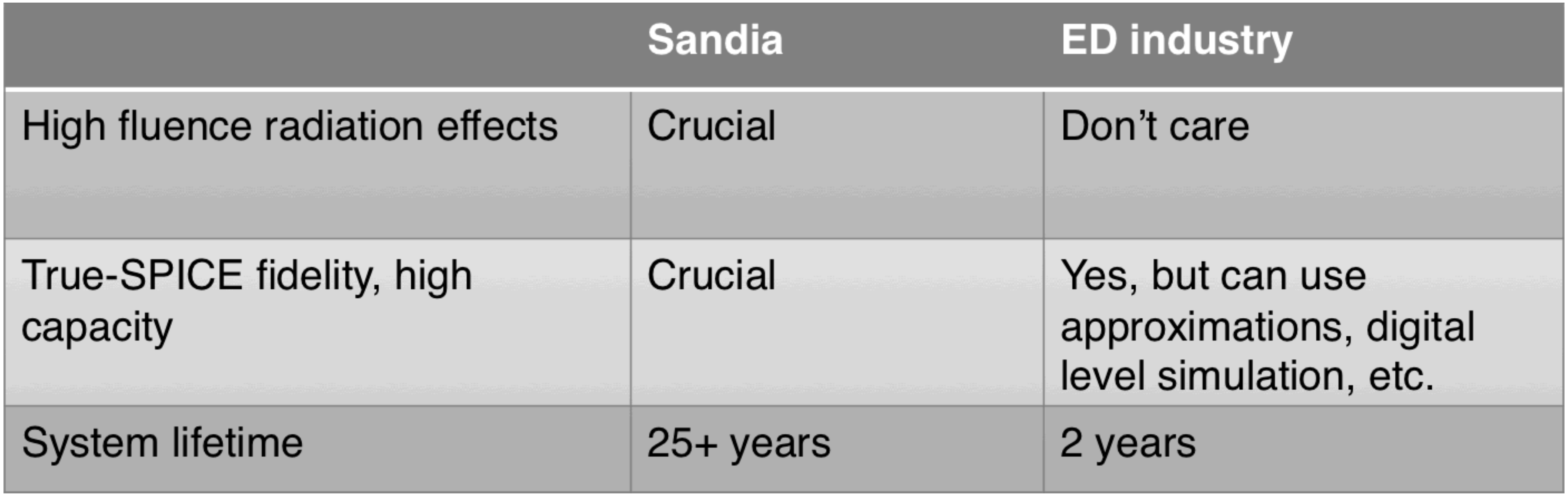
Sandia-specific enhancements:

- Radiation effects (photocurrent and neutron damage)
- Enhanced iterative linear solvers
- Improved time integration schemes
- External code coupling (Charon, Alegra, Emphasis)
- Additional industry-standard models (e.g. VBIC, FBH HBT, PSP MOSFET, EKV MOSFET)
- Massively Parallel
- Extreme size problems at SPICE-level fidelity

Differences from SPICE

- Simulation engine ONLY:
  - No GUI.
  - No schematic capture.
  - No built-in graphical displays of results.
  - Text-format netlist input only.
- Options, analysis, and output statements differ from many SPICE variants.
- Replaces only some of normal simulation workflow: you use your own schematic editor/netlist generator, data analysis software.

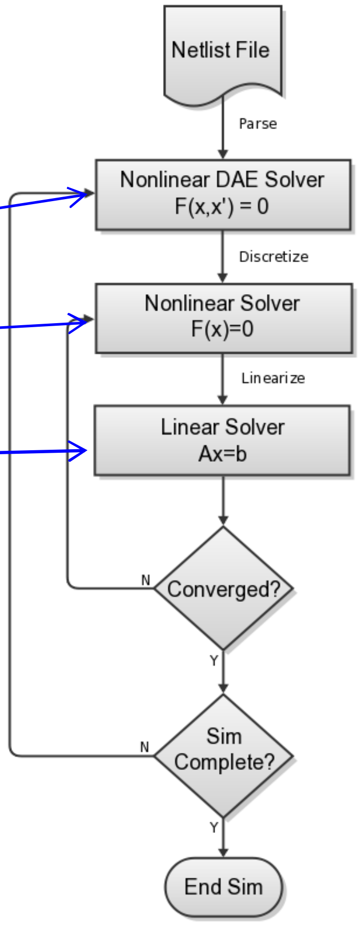
Contrast to Industry Needs



	Sandia	ED industry
High fluence radiation effects	Crucial	Don’t care
True-SPICE fidelity, high capacity	Crucial	Yes, but can use approximations, digital level simulation, etc.
System lifetime	25+ years	2 years

Transistor-Level 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 hasphases
  - Compute starting point (DCOP)
  - Start analysis (transient)
  - Sparse linear algebra / solvers
    - Linchpin of scalable and robust performance



Comparable Parallel Simulation Approaches

Xyce is a distributed memory, MPI-based analog circuit simulator started in 1999

- Hybrid parallelism (MPI+threads) depending on choice of linear solver

Many commercial simulators have incorporated parallelism

- Access to multi-processor / multi-core desktops
- Multithreading key portions, newer simulators redesigned from the ground up

Multi-Algorithm Parallel Circuit Simulation (MAPS)

[X. Ye, W. Dong, P. Li, S. Nassif]

- Multiple numerical integration methods with synchronization

WavePipe

[W. Dong, P. Li, X. Ye]

- Multi-core, shared-memory simulator
- Emulate hardware pipelining to expedite time integration

Domain-decomposition Parallel Simulation

[H. Peng and C.K. Cheng]

- Divides circuit into linear and non-linear components (subdomains)

Trilinos

Trilinos is an evolving framework to support large-scale simulation codes:

- Fundamental atomic unit is a *package*
- Includes core set of vector, graph and matrix classes (Epetra/Tpetra packages)
- Provides a common abstract solver API (Thyra package)
- Provides a ready-made package infrastructure:
  - Source code management (cvs, bonsai)
  - Build tools (cmake)
  - Automated regression testing (queue directories within repository)
  - Communication tools (mailman mail lists)
- Specifies requirements and suggested practices to address ASC SQA/SQE requirements

Trilinos allows the separation of efforts:

- Efforts best done at the Trilinos level (useful to most or all packages)
- Efforts best done at a package level (peculiar or important to a package)
- **Allows package developers to focus only on things that are unique to their package**

Xyce Trilinos Package Summary

	Objective	Package(s)
Discretizations	Spatial Discretizations (FEM,FV,FD)	Intrepid
	Time Integration	Rythmos
Methods	Automatic Differentiation	Sacado
	Mortar Methods	Moertel
Core	Linear algebra objects	Epetra, Jpetra, Tpetra
	Abstract interfaces	Thyra, Stratimikos, RTOp
	Load Balancing	Zoltan, Isorropia
	"Skins"	PyTrilinos, WebTrilinos, Star-P, ForTrilinos
	C++ utilities, (some) I/O	Teuchos, EpetraExt, Kokkos, Triutils
Solvers	Iterative (Krylov) linear solvers	AztecOO, Belos, Komplex
	Direct sparse linear solvers	Amesos
	Hybrid direct-iterative solvers	ShyLU
	Direct dense linear solvers	Epetra, Teuchos, Pliris
	Iterative eigenvalue solvers	Anasazi
	ILU-type preconditioners	AztecOO, IFPACK, TIFPACK
	Multilevel preconditioners	ML, CLAPS
	Block preconditioners	Meros
	Nonlinear system solvers	NOX, LOCA
	Optimization (SAND)	MOOCHO, Aristos

ICCAD
2009 CMS
Workshop

A New 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: $S = G - R * D^{-1}C.$

Achieving Scalability and Robustness within Xyce

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

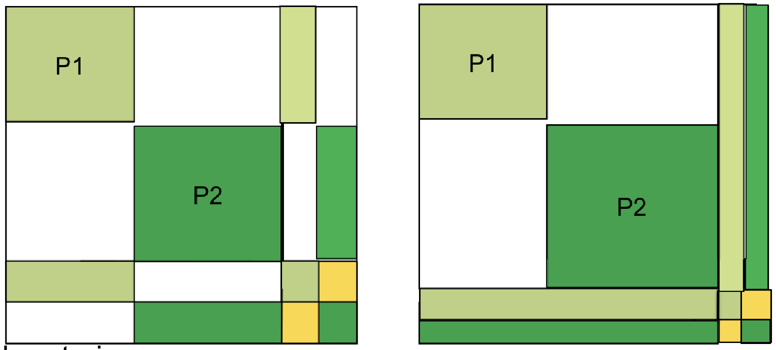
ShyLU is used as a stand-alone solver in Xyce

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 Solvers: Recent Comparisons

Recently, several parallel hybrid solvers based on Schur complements have been developed:

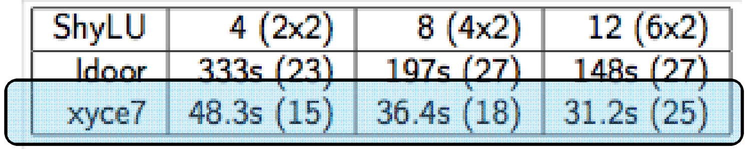
- **HIPS** (Gaidamour, Henon)
- MaPhys (Giraud, Haidar, et al.)
- PDSlin (Li, Ng, Yamazaki)
- **ShyLU** (Rajamanickam, Boman, Heroux)

They differ in many ways, e.g, how they approximate Schur complements and how they partition/reorder the matrix.

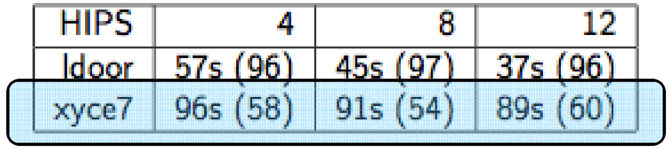
Numerical tests comparing ShyLU and HIPS were run a 12-core (dual hex) workstation

HIPS – MPI only

ShyLU – MPI+threads



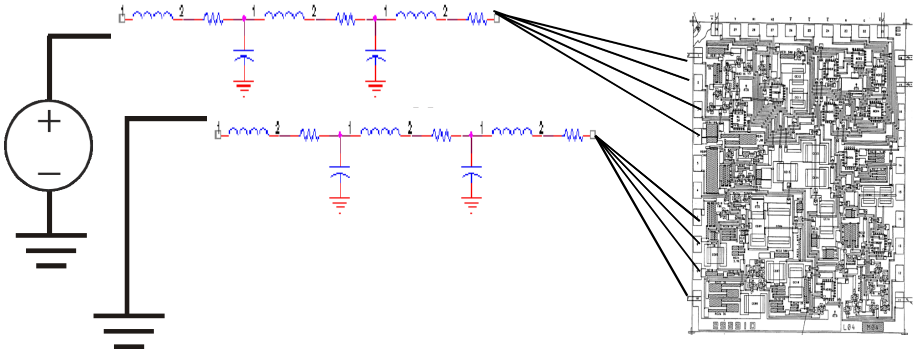
ShyLU	4 (2x2)	8 (4x2)	12 (6x2)
Idoor	333s (23)	197s (27)	148s (27)
xyce7	48.3s (15)	36.4s (18)	31.2s (25)



HIPS	4	8	12
Idoor	57s (96)	45s (97)	37s (96)
xyce7	96s (58)	91s (54)	89s (60)

Challenge: Power Node Parasitics

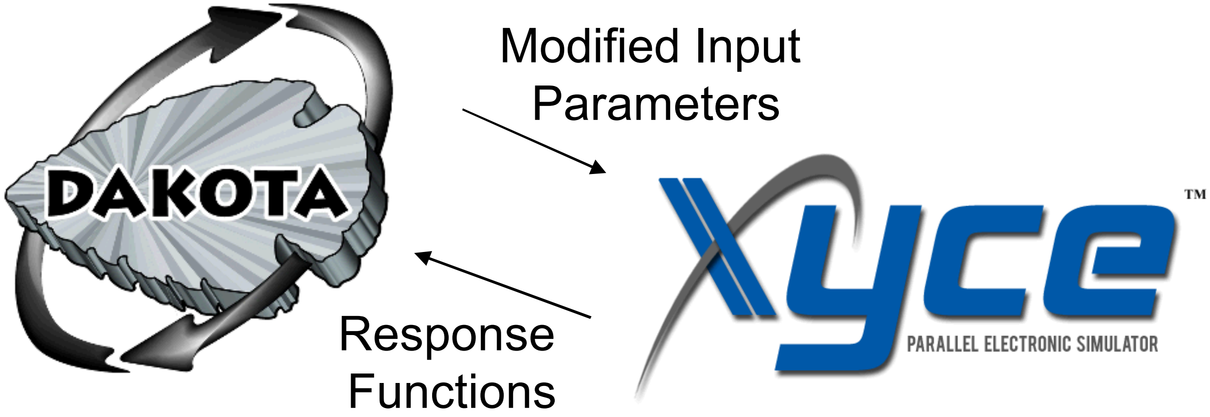
- Intra-circuit parasitics are ok for most simulations.
- Parasitic elements on the power and/or ground nodes of large digital circuits are a problem:
  - Tend to be highly connected.
  - High impact on solver difficulties.
  - Break “singleton removal”



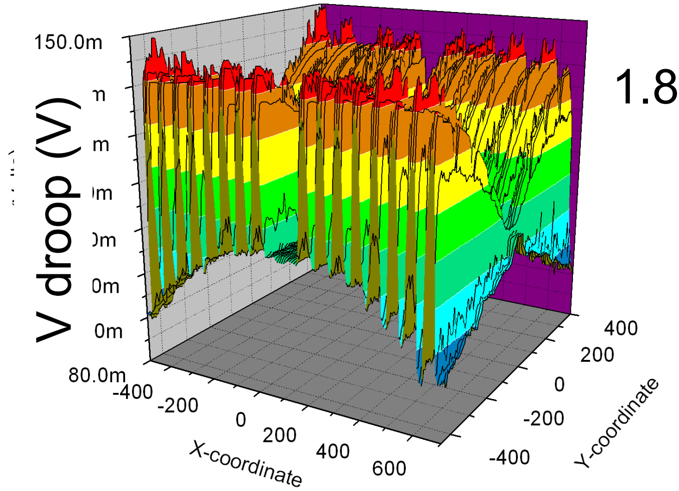
Optimization and UQ

Xyce couples directly and indirectly with Dakota

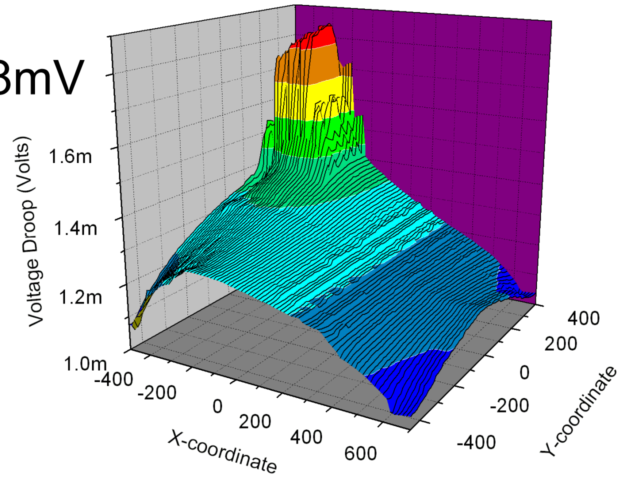
- Parameter studies
- Model parameter extractions
- Uncertainity Quantificaiton



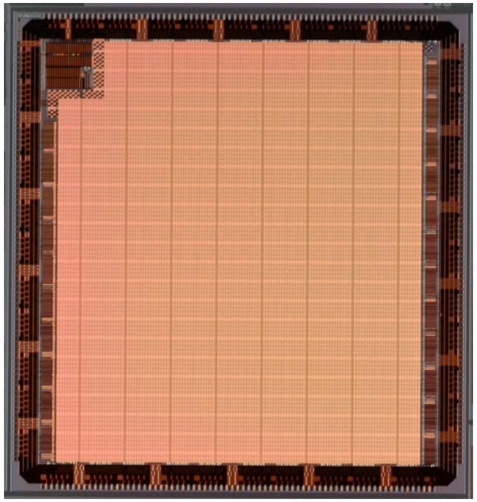
Optimization



Worst case



Best case



Xyce simulations predict state-dependent photocurrent response.

Photocurrent response function of design choices:
Metal layer widths and constraints
power routing.

DAKOTA + Xyce