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
Parallel Circuit Simulator

- **Xyce**: Massively Parallel circuit simulator:
  - Distributed Memory Parallel (MPI-based)
  - Unique solver algorithms
  - SPICE-Compatible
  - Industry standard models (BSIM, PSP, EKV, VBIC, etc)

- **Analysis types**
  - DC, TRAN, AC
  - Harmonic Balance (HB)
  - Multi-time PDE (MPDE)
  - Model order reduction (MOR)
  - Direct and Adjoint sensitivity analysis
  - Uncertainty quantification (UQ) via Dakota.

- **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
  • Transient Photocurrent
  • Neutron Damage

Circuit response
  • Devices
  • Integrated circuit
  • Circuit Board
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.
SNL has six core technical capabilities

- Microelectronics and Photonics
- Materials Science & Technology
- Computational & Informational Sciences
- Engineering Sciences
- Pulsed Power
- Bioscience

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

- Gordon Bell Prize
- R&D 100 Parallel Software
- Patent Meshing
- R&D 100 Dense Solvers
- World Record Teraflops
- Gordon Bell Prize
- R&D 100 Storage
- World Record
- SC96 Gold Medal Networking
- R&D 100
- R&D 100 Trilinos
- R&D 100 Allocators
- Mannheim SuParCup
- Patent Data Mining
- R&D 100 3D-Touch
- Fernbach Award
- R&D 100 Salvo
- Patent Paving
- Patent Decimation
- Eric Keiter, Sandia National Laboratories
2013 MOS-AK Workshop, Washington DC
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

Dakota capabilities:
- LHS sampling (parallel)
- Importance sampling
- Failure analysis methods
- Stochastic collocation
- Polynomial chaos
- Model calibration/extraction
- Gradient-based optimization

Circuit Simulator Application

Solver Library

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

- 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.

100K Transistor IC Problem
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
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 older CMOS memory circuits
  • Circuits with parasitics are more challenging
  • Modern MOS models (w/gate leakage) break it.

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

- W. Bomhof and H.A. van der Vorst [NLAA, 2000]

<table>
<thead>
<tr>
<th>Preconditioning Method</th>
<th>Residual</th>
<th>GMRES Iters</th>
<th>Solver Time (s.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local AMD ILUT ParMETIS</td>
<td>3.43e-01 (FAIL)</td>
<td>500</td>
<td>302.573</td>
</tr>
<tr>
<td>BTF Block Jacobi Hypergraph</td>
<td>3.47e-10</td>
<td>3</td>
<td>0.139</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>TABLE III</th>
<th>Comparison of total linear solve time (sec.) of various sparse direct solvers for our test circuits; (-) indicates simulation failed to complete.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ckt1</td>
</tr>
<tr>
<td>KLU</td>
<td>80.8</td>
</tr>
<tr>
<td>PARDISO (16)</td>
<td>128.6</td>
</tr>
<tr>
<td>SuperLU</td>
<td>-</td>
</tr>
<tr>
<td>SuperLU_Dist (16)</td>
<td>-</td>
</tr>
</tbody>
</table>

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

then ~ 2001
PSpice circuit model:
~300 devices

rad. model:
empirical/behavioral

2006
Xyce circuit model:
300K+ devices

rad. model:
physics-based/built-in

2012
Xyce circuit model:
40M+ devices
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.

Run admsXyce
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, but the MEXTRAM verilog can be (and has been) processed by our ADMS back-end).
- MVS Silicon virtual source model from nanohub.com (processed last week in ADMS, not really tested yet)
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.
- Dynamic linking: this is on our roadmap, but hasn’t been done yet.
Using ModSpec in Xyce

(extended) Verilog-A model

ModSpec Model
\textbf{(C++ API)}

\begin{itemize}
  \item automatic translator
  \item \textbf{compile standalone}
  \item key: standardized low-level interface
\end{itemize}

\begin{itemize}
  \item \textbf{binary release possible (IP protection)}
  \item \textbf{test immediately (standalone)}
  \item \textbf{speed near native/hand-coded implementation (compiled C++ code)}
\end{itemize}

\textbf{model supported in \textit{Xyce}}

\textbf{.so libraries (dynamically loadable)}

via ModSpec C++ API support
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
Thanks!