Xyce & miniXyce

Karthik Aadithya

Eric Keiter

Heidi Thornquist

Electrical & Microsystems Modeling

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company,
for the United States Department of Energy under contract DE-AC04-94AL85000.

Sandia National Laboratories

Summary

- Xyce
  - Circuit simulator overview
  - Communication / computations
- miniXyce
  - Simple linear device simulator
  - Performance comparison
- Conclusions / Future directions

Xyce

- Parallel SPICE-style circuit simulation code
- MPI-based
  - localized threading
- Third party libraries
  - Trilinos
  - SuperLU/UMFPACK
- SLOC: 370K
  - Mostly devices

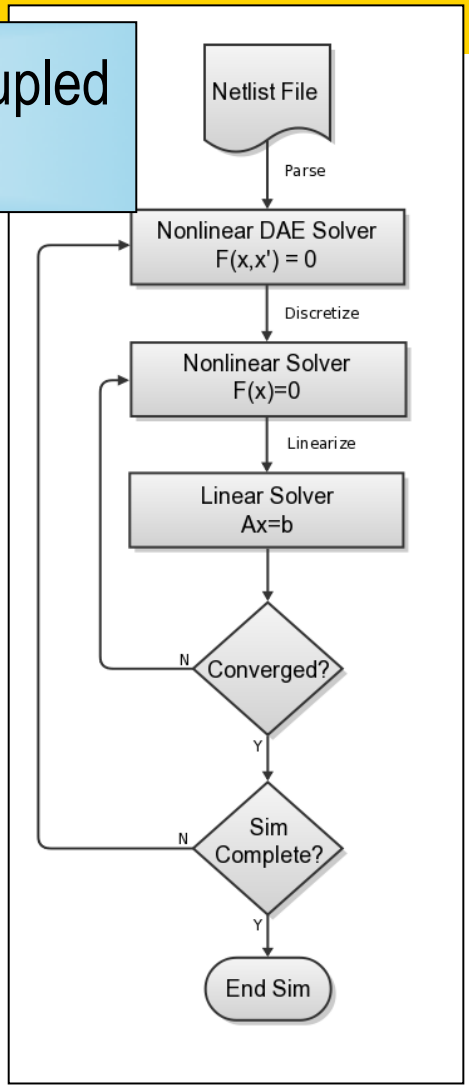
```
SLOC    Directory       SLOC-by-Language (Sorted)
271597  DeviceModelPKG  cpp=271443,lex=154
25613   IOInterfacePKG  cpp=25613
15531   LinearAlgebraServicesPKG ansic=8941,cpp=6590
14172   NonlinearSolverPKG cpp=14172
9311    UtilityPKG      cpp=9311
8986    TimeIntegrationPKG cpp=8986
7550    TopoManagerPKG  cpp=7550
6710    AnalysisPKG     cpp=6710
5754    MultiTimePDEPKG cpp=5754
5478    ParallelDistPKG cpp=5478
4473    test            cpp=4473
1481    CircuitPKG      cpp=1481
1033    ErrorHandlingPKG cpp=1033
825     LoaderServicesPKG cpp=825
581     DakotaLinkPKG   cpp=581
157     CircuitStatePKG cpp=157


Totals grouped by language (dominant language first):
cpp:         370157 (97.60%)
ansic:         8941 (2.36%)
lex:            154 (0.04%)
```

General Circuit Simulation Flow

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

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



- Netlist parser
- Nonlinear DAE solver
  - Nonlinear solver
  - Linear solver

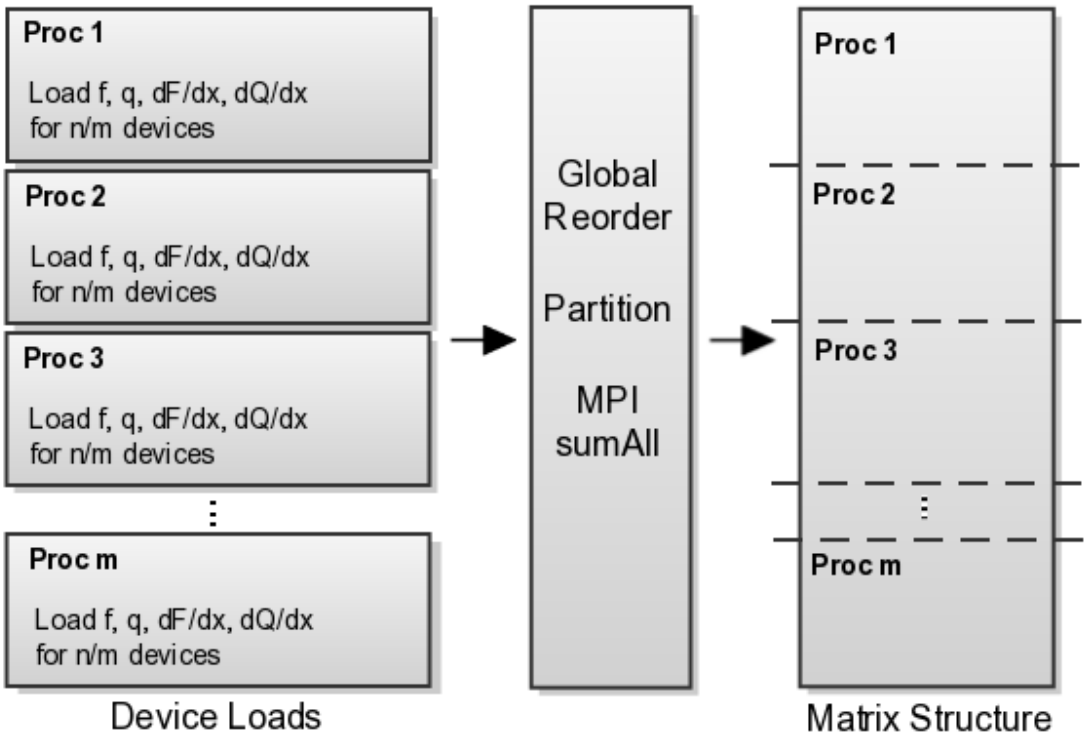
Netlist Parser

- Design for flexibility:
  - On some systems, some nodes may not have I/O
  - On clusters, nodes may have completely independent file systems.
  - Most large netlists are very hierarchical, which makes parallel I/O tricky (but not impossible).
- Lessons learned:
  - Original Xyce parser would do everything on processor 0, and then distribute to other processors.
  - For larger problems, this approach ran out of memory on processor 0.
- Parser redesign:
  - Minimal processing on proc 0.
  - Each processor streams in its own portion of the circuit.

Netlist Parser

- Netlist is streamed in on processor 0, multiple passes.
- Pass 1:
  - Global “Dot” statements (.TRAN, .OPTION, .PRINT, etc) broadcast to all procs.
  - Symbolic flatting, including total device count. Get D/N.
  - File pointers determined: .SUBCKT locations. Most subckt info left in netlist file, not stored in memory.
- Pass 2:
  - On proc 0, stream in file in blocks of lines at a time.
  - Based on previous symbolic flattening, resolve names (node, device, model)
  - MPI_send resolved devices as raw character buffers to next processor.
  - Once the current “send” processor has D/N devices, move on to next.
  - Each proc allocates devices as they are received, and owned only by that processor.
- Parsing mostly operation-serial.

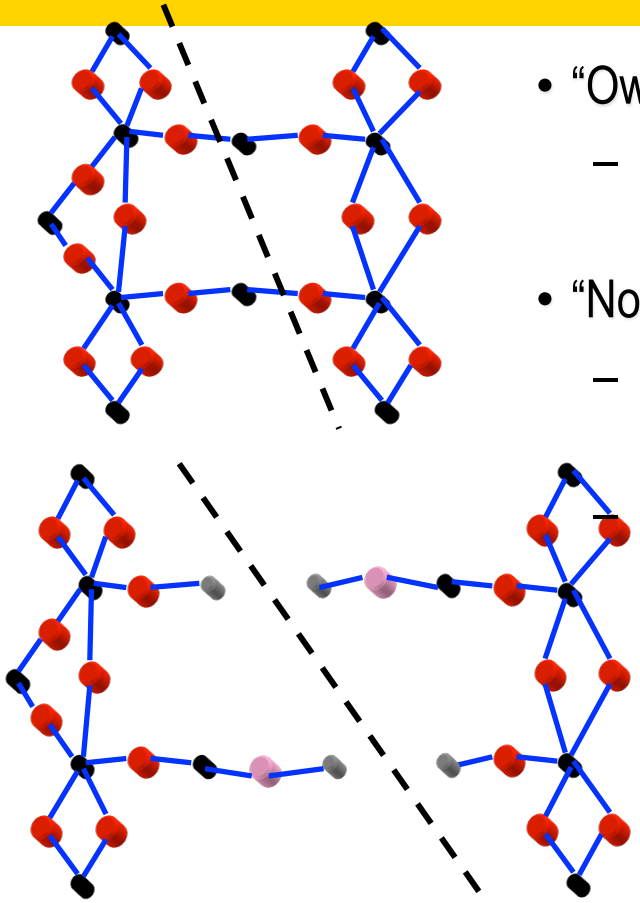
Parallel Approach for Nested Solver Loop



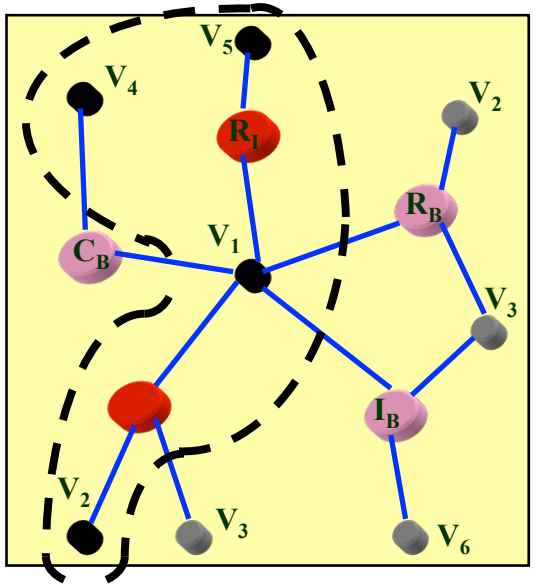
Device Evaluation Phase

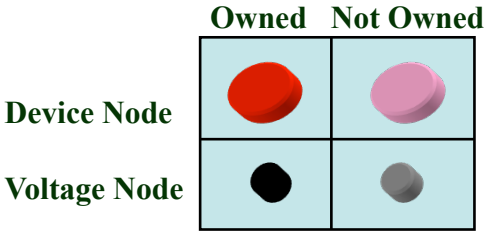
- Balanced by taking into account only the computational work required
- Can take advantage of coarse and fine scale parallelism
  - Partition devices across processors (coarse)
  - Devices of the same type are evaluated at the same time (fine)

Device vs. Node “Ghosting”



- “Owned”/Internal node
  - processor loads associated rows
- “Not Owned”/External node
  - Dev-node: Load to V-node rows
  - V-node: Reference for nonlocal data
    - Requires global communication to update distribute shared solution vector data





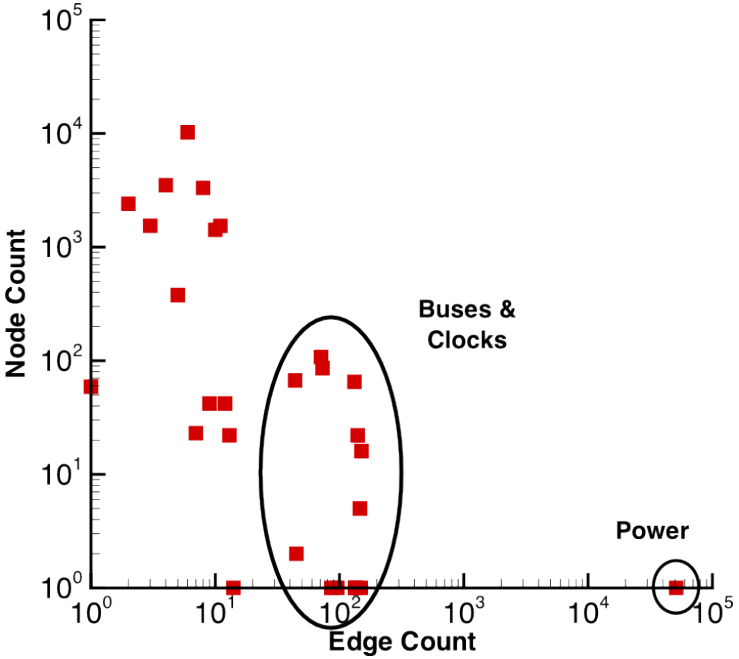
Linear Solution Phase

The linear systems can be challenging to solve because they are heavily influenced by:

- Network Connectivity
  - Hierarchical structure rather than spatial topology
  - Densely connected nodes: O(n)
- Badly Scaled DAEs
  - Compact models designed by engineers, not numerical analysts!
  - DCOP matrices are often ill-conditioned
- Non-Symmetric
  - Heterogeneous matrix structure
  - Not elliptic and/or globally SPD

Network Connectivity

Singleton Removal



Row Singleton: pre-process

$$\begin{bmatrix} & & a_{1j} & & \\ & & a_{2j} & & \\ & & \vdots & & \\ & & \vdots & & \\ 0 & \cdots & 0 & a_{ij} & 0 & \cdots & 0 \\ & & \vdots & & \\ & & a_{nj} & & \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ \vdots \\ x_j \\ \vdots \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ \vdots \\ b_i \\ \vdots \\ b_n \end{bmatrix}$$

$$\Rightarrow x_j = b_i / a_{ij}$$

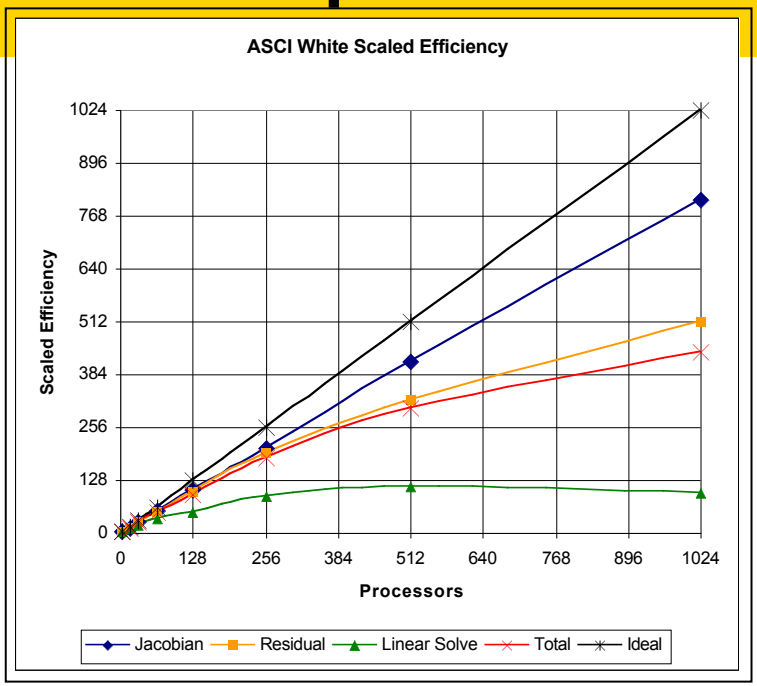
Column Singleton: post-process

$$\begin{bmatrix} & & 0 & & \\ & & 0 & & \\ & & \vdots & & \\ & & \vdots & & \\ a_{i1} & \cdots & \cdots & a_{ij} & \cdots & \cdots & a_{in} \\ & & \vdots & & \\ & & 0 & & \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ \vdots \\ x_j \\ \vdots \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ \vdots \\ b_i \\ \vdots \\ b_n \end{bmatrix}$$

$$\Rightarrow x_j = \left(b_i - \sum_{k \neq j} a_{ik} x_k \right) / a_{ij}$$

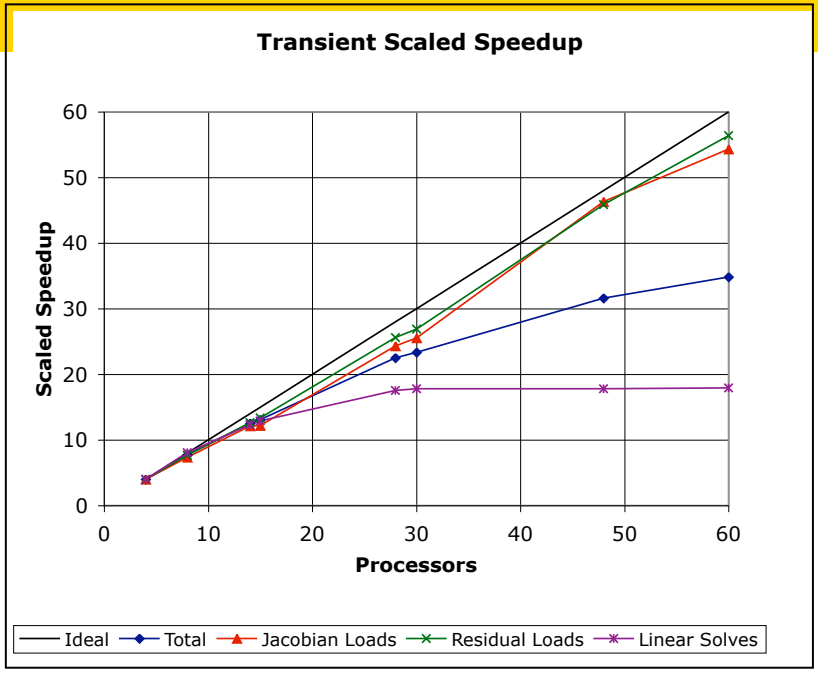
- Connectivity:
  - Most nodes very low connectivity -> sparse matrix
  - Power node generates very dense row (~0.9*N)
  - Bus lines and clock paths generate order of magnitude increases in bandwidth

Computational Performance



Transmission line scaling

variable problem size



ASIC scaling

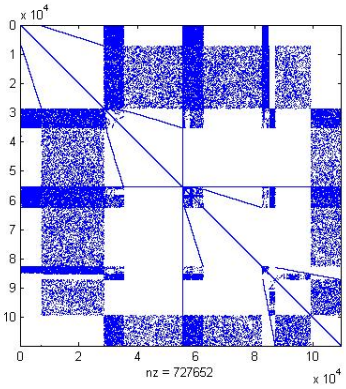
fixed problem size

- Transmission line (max size = 14 million devices).
- ASIC scaling on the right. (much harder problem)
- For both problems, roll off occurs in the linear solve phase.

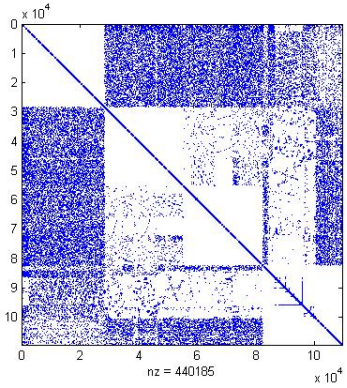
Computational Performance

100K Transistor IC Problem

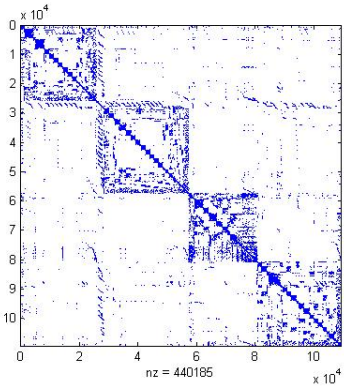
Original



ParMETIS+AMD

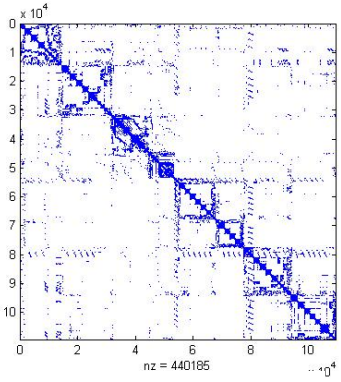


BTF+Hypergraph



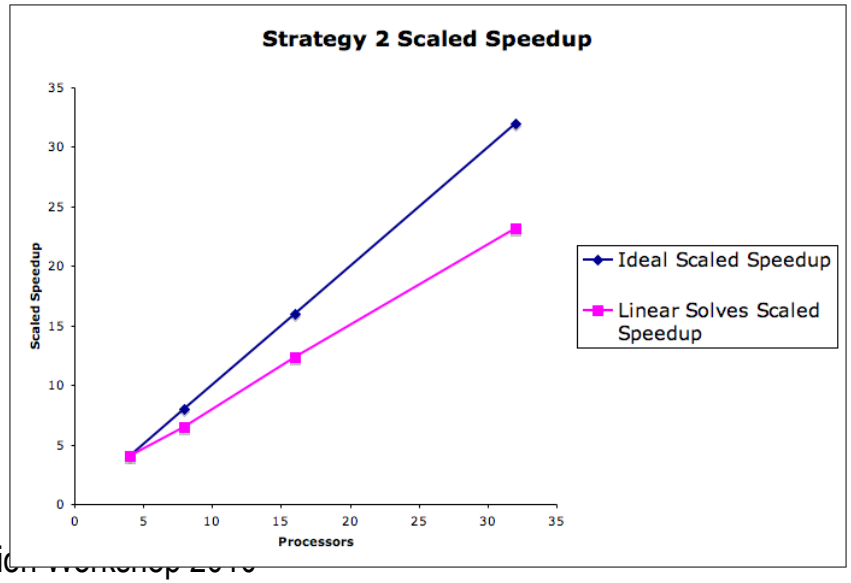
4 processors

BTF+Hypergraph



8 processors

Strategy	Method	Residual	GMRES Iters	Solver Time (seconds)
1	Local AMD ILUT ParMETIS	3.425e-01	500	302.573
2	BTF KLU Hypergraph	3.473e-10	3	0.139



miniXyce

- Small kernels emulating vital computation / communication for circuit simulation
- First attempt: simple simulation of RC ladder
  - no parser, serial, CG solver, one voltage source & type
- Second attempt: simple linear circuit simulator
  - any circuit with R,L,C components
  - basic netlist parser, single pass
  - multiple voltage/current sources & types
  - GMRES solver, no preconditioner

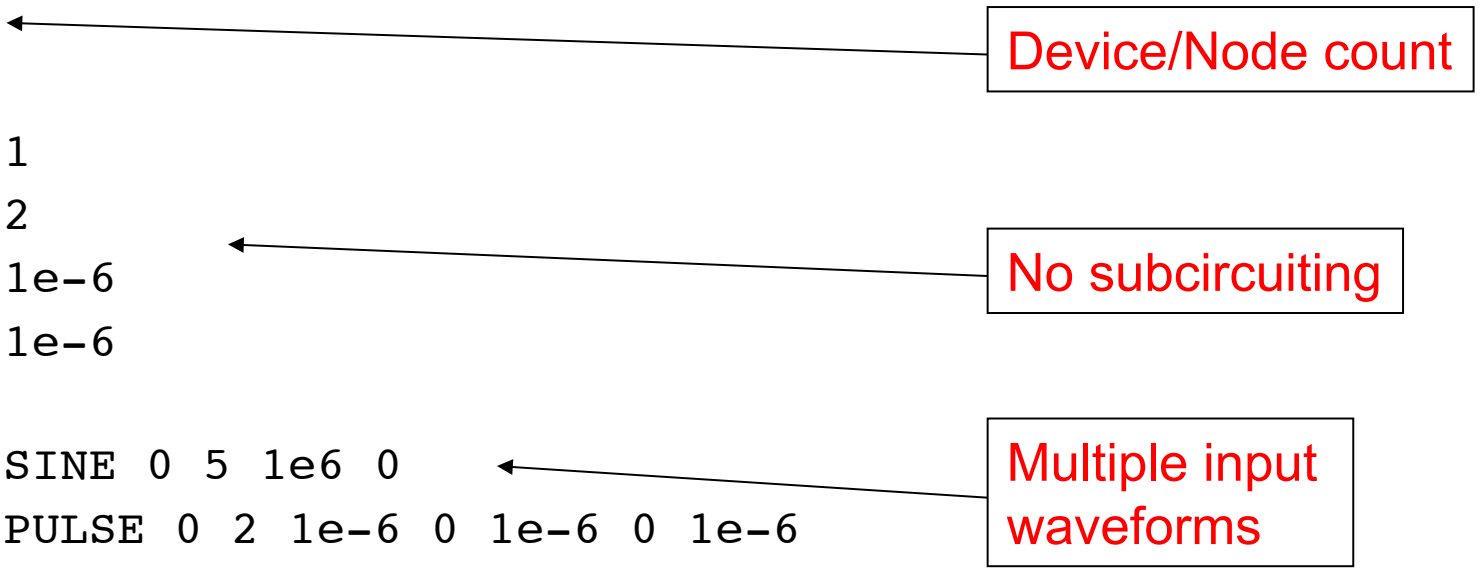
Netlist Parser

```
% simple circuit 5
% includes all the components {R,L,C,V,I}

3 1 1

R1 1 2 1
R2 2 0 2
L1 2 3 1e-6
C1 3 0 1e-6

V1 1 0 SINE 0 5 1e6 0
I1 2 0 PULSE 0 2 1e-6 0 1e-6 0 1e-6
```



Simulation Parameters

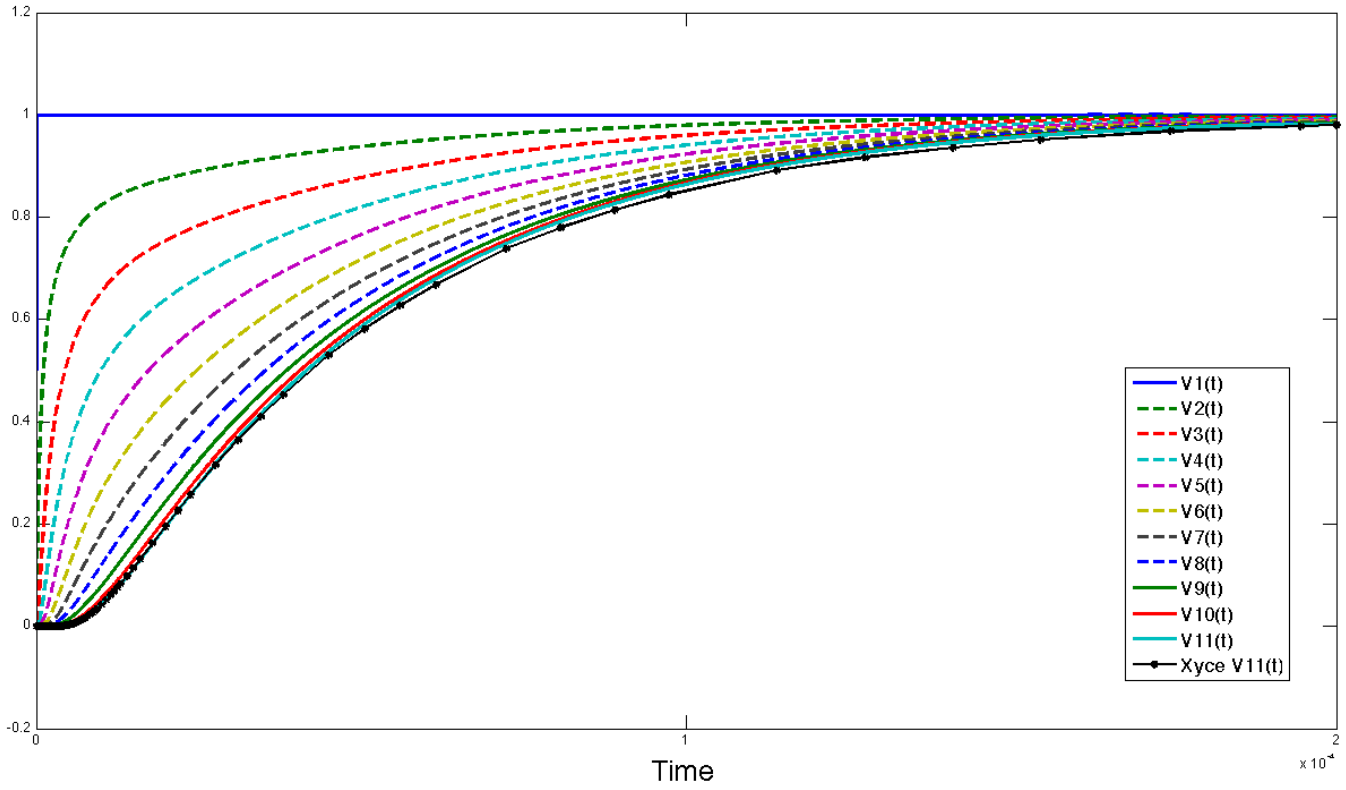
- Command-line and file inputted parameters

```
circuit = cir5.net
t_start = 0
t_step = 1e-8
t_stop = 10e-6
tol = 1e-06
k = 10
```

- Each run outputs `last_used_parms.txt` file

```
mpirun -np 4 parallel_mx --prev
```

RC Ladder
Simulation Verification



… ok, this looks good, so what's next?

Conclusions / Future Work

- Compare Xyce & miniXyce
  - scale up problems & processors
  - are simulator profiles similar?
  - but this is not indicative of reality …
- Individual kernels for analyzing:
  - parsing
  - device evaluation / matrix load kernel
  - graph analysis / linear solver / preconditioner