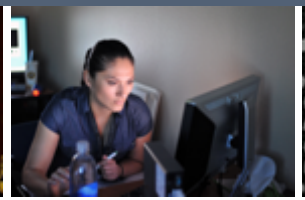
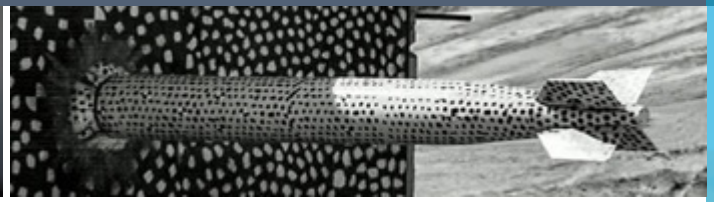
SAND2021-12272PE

Polynomial Chaos Expansion (PCE) Methods with the Xyce Circuit simulator





Dr. Eric Keiter, Sandia National Laboratories

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and 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.

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.

Outline

- Part 1: Xyce open source circuit simulator overview

  https://xyce.sandia.gov

  https://github.com/xyce

- Part 2: Polynomial Chaos methods in Xyce

The Xyce™ Parallel Electronic Simulator Analog Circuit Simulator

- SPICE-Compatible syntax
  - Berkeley 3f5
- **Distributed Memory Parallel** (MPI-based)
- Unique solver algorithms
- Industry standard models

https://xyce.sandia.gov

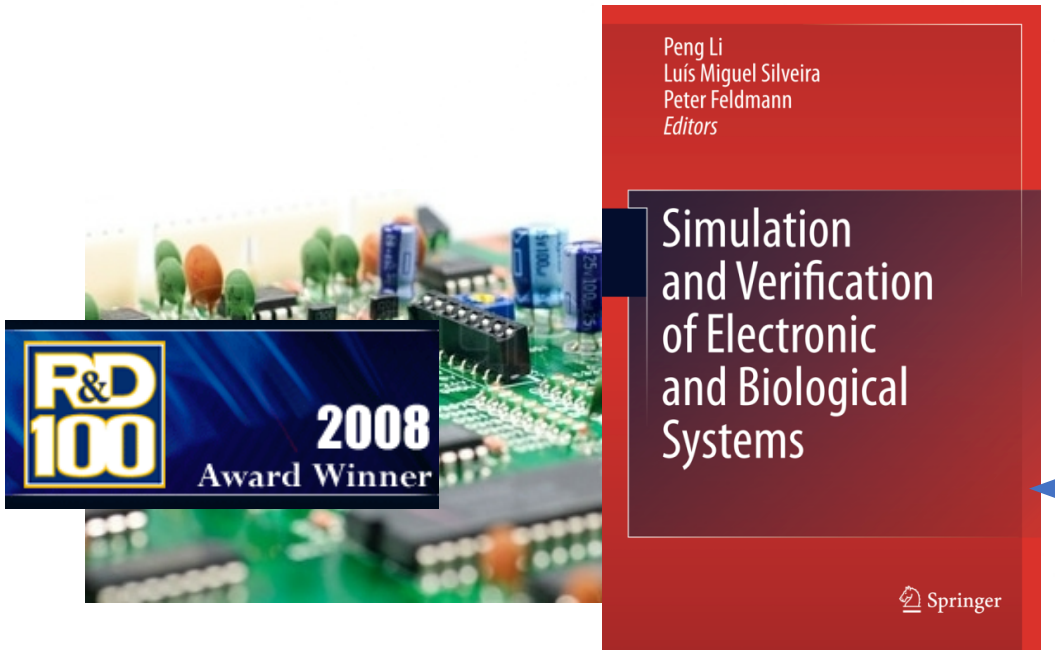
https://github.com/xyce

Open Source, GPLv3

- Since September of 2013 (Xyce 6.0)

Xyce Release v7.3

- May, 2021 (possibly today!)
- 29th major release



Keiter, et al., "Parallel Transistor-Level Circuit Simulation"

Sandia National Laboratories

Why Open Source?

- Foster external collaboration
- Feedback from wider community
- Taxpayer funded, so encouraged to open source
- Some of our funding requires it

- First open source release, v6.0
- November 5, 2013.
- GPL license v3.0
- Source and binary downloads available
- Next release (v7.3) ~May 2021.

https://xyce.sandia.gov

https://github.com/xyce

Xyce Capabilities

Typical

- DC, Transient, AC, Noise
  - .DC, .TRAN, .NOISE, .AC (and .STEP)
- Post Processing:
  - Fourier transform of transient output (.FOUR)
  - Post-simulation calculation of simulation metrics (.MEASURE)
- Output (.PRINT)
  - Text Files (tab or comma delimited)
  - Probe
  - Gnuplot, TecPlot, RAW
- Analog Behavioral Modeling
- Expressions, functions, parameterizations…

Others

Harmonic Balance Analysis (.HB)

- Steady state solution of nonlinear circuits in the frequency domain

Random Sampling Analysis

- Executes the primary analysis (.DC, .AC, .TRAN, etc.) inside a loop over randomly distributed parameters

Sensitivities

- Computes sensitivities for a user-specified objective function with respect to a user-specified list of circuit parameters ($\partial O / \partial p \ldots$)
- DC or Transient
- E.g., an output voltage's dependence on a capacitance

Polynomial Chaos methods (new!)

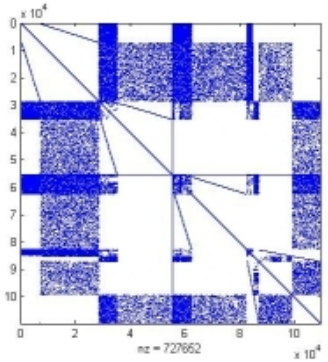
- Quadrature
- Regression

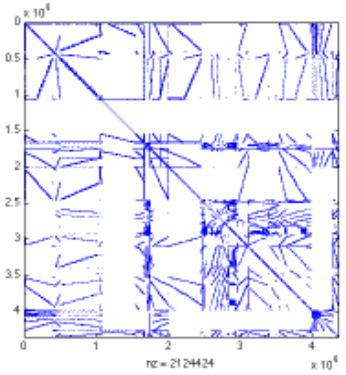
Parallel Circuit Simulation Challenges

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

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

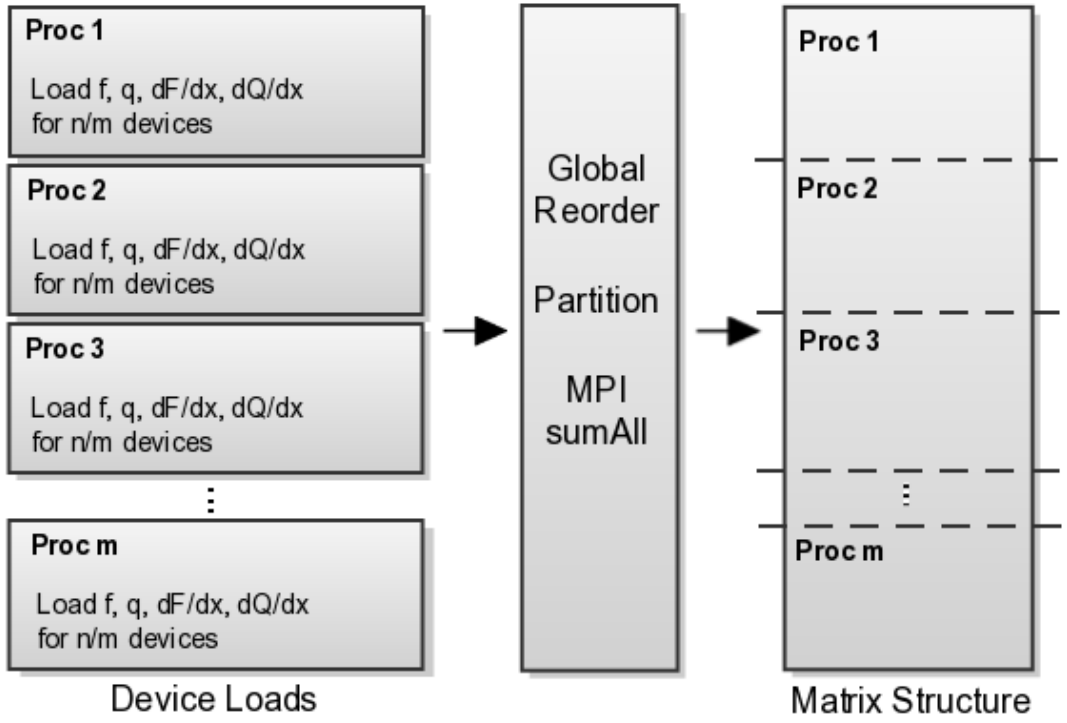
- Network Connectivity
  - Hierarchical structure rather than spatial topology
  - Densely connected nodes: O(n)
- Badly Scaled DAEs
  - Compact models designed by engineers, not numerical analysts!
  - Steady-state (DCOP) matrices are often ill-conditioned
- Non-Symmetric Matrices
- Load Balancing vs. Matrix Partitioning
  - Balancing cost of loading Jacobian values unrelated to matrix partitioning for solves
- Strong scaling and robustness is the key challenge!





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

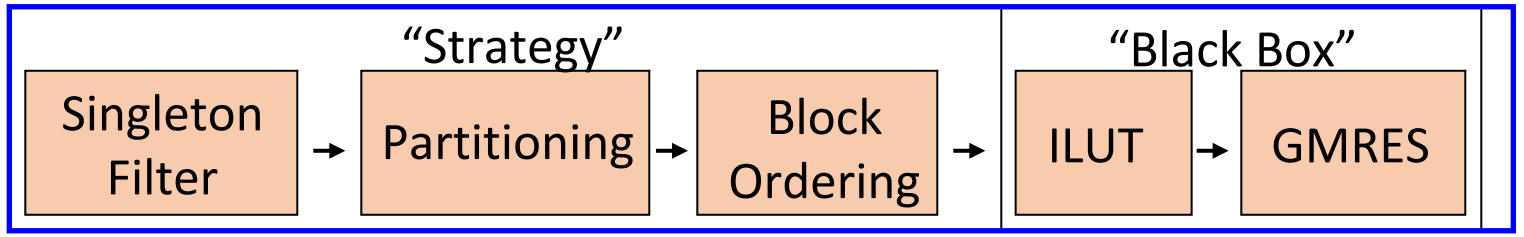


Parallel Iterative Matrix Solvers

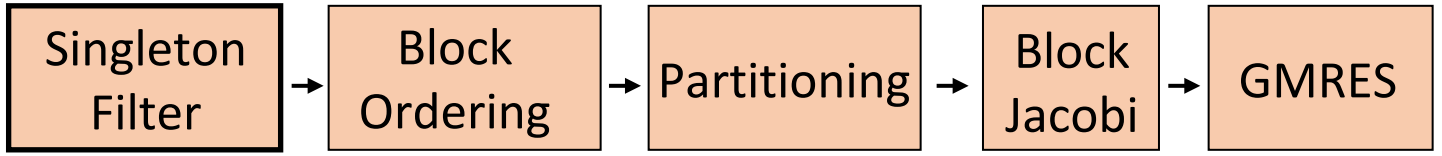
- Iterative methods scale much better than direct methods, but circuit matrices are difficult:
  - Network Connectivity: Hierarchical Structure, Densely Connected Nodes: O(n)
  - Badly Scaled DAEs
  - Non-Symmetric: Not Elliptic and/or SPD
- Black box methods won't work for circuits!
  - Need comprehensive strategy

Strategy	Precond	N	Total Cuts	Condition #	GMRES Iters	Solve Time
Black Box	ILUT	1220	~1000	3.00E+05	500	4.7
Strategy 1	SF+ILUT+RCM +ZOLTAN	1054	68	1.00E+04	127	0.43

Strategy 1

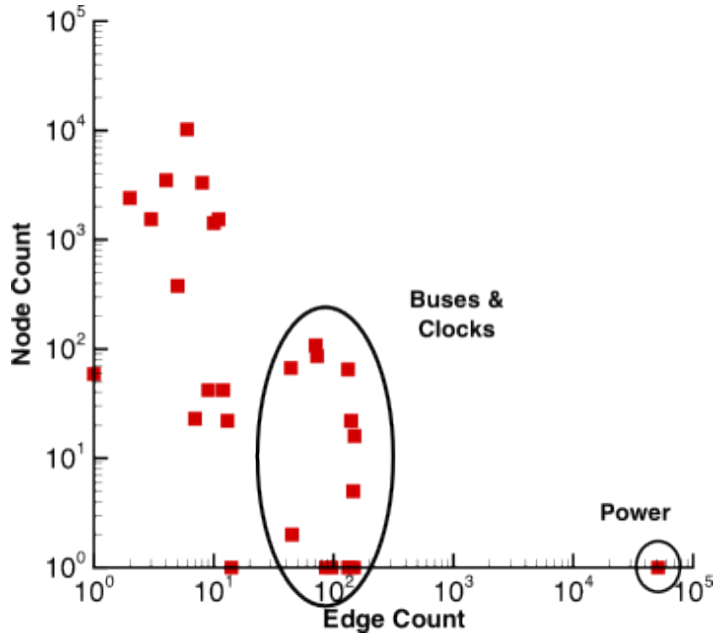


Strategy 2



"A Parallel Preconditioning Strategy for Efficient Transistor-Level Circuit Simulation", Thornquist, Keiter, et al, ICCAD 2009

Singleton Filtering



Row Singleton: Pre-Process

$$\begin{bmatrix} & a_{1j} & \\ & a_{2j} & \\ & \mathrm{M} & \\ & \mathrm{M} & \\ 0 \;\mathrm{L}\; 0 & a_{ij} & 0 \;\mathrm{L}\; 0 \\ & \mathrm{M} & \\ & a_{nj} & \end{bmatrix} \begin{bmatrix} x_1 \\ \mathrm{M} \\ \mathrm{M} \\ x_j \\ \mathrm{M} \\ \mathrm{M} \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \mathrm{M} \\ \mathrm{M} \\ \mathrm{M} \\ b_i \\ \mathrm{M} \\ b_n \end{bmatrix} \Rightarrow x_j = b_i / a_{ij}$$

Column Singleton: Post-Process

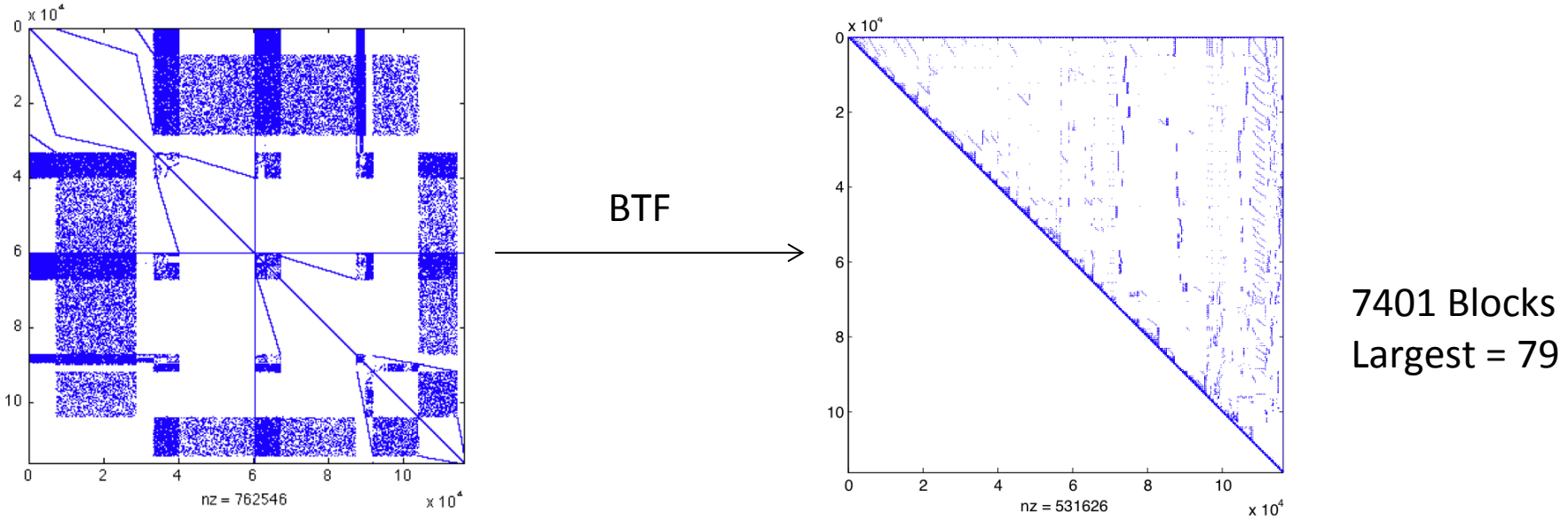
$$\begin{bmatrix} & 0 & \\ & 0 & \\ & \mathrm{M} & \\ & 0 & \\ a_{i1} \;\mathrm{L}\; & a_{ij} & \;\mathrm{L}\; a_{in} \\ & 0 & \\ & 0 & \end{bmatrix} \begin{bmatrix} x_1 \\ \mathrm{M} \\ \mathrm{M} \\ x_j \\ \mathrm{M} \\ \mathrm{M} \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \mathrm{M} \\ \mathrm{M} \\ \mathrm{M} \\ b_i \\ \mathrm{M} \\ b_n \end{bmatrix} \Rightarrow x_j = (b_i - \sum_{k \neq j} a_{ik} x_k) / a_{ij}$$

- 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

Singleton Filter → Block Ordering → Partitioning → Block Jacobi → GMRES

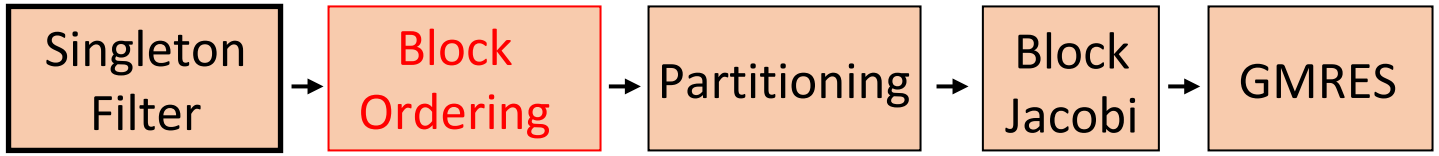
"A Parallel Preconditioning Strategy for Efficient Transistor-Level Circuit Simulation", Thornquist, Keiter, et al, ICCAD 2009

Reordering: Block Triangular Form (BTF)



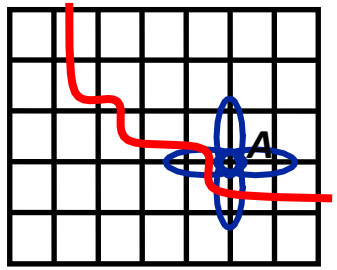
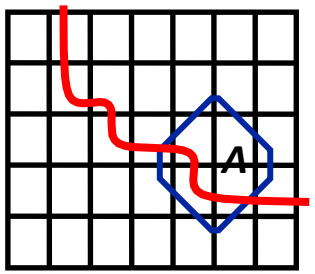
7401 Blocks
Largest = 79

- David Day (SNL) showed that Strongly Connected Components can result in a Block Triangular Form
- BTF structure exists in many, but not all, circuit problems.
  - Common in CMOS Memory circuits
- BTF Algorithm: $O(n_b \cdot s_b^3 + n_b^2 \cdot s_b^2)$
- Benefits both Direct and Preconditioned Iterative Methods
- Used by Tim Davis's KLU in Trilinos/AMESOS (The "Clark Kent" of Direct Solvers)



"A Parallel Preconditioning Strategy for Efficient Transistor-Level Circuit Simulation", Thornquist, Keiter, et al, ICCAD 2009

Hypergraph Partitioning

Graph Partitioning Kernighan, Lin, Schweikert, Fiduccia, Mattheyes, Pothen, Simon, Hendrickson, Leland, Kumar, Karypis, et al.	Hypergraph Partitioning Kernighan, Alpert, Kahng, Hauck, Borriello, Aykanat, Çatalyürek, Karypis, et al.
Edges: two vertices.	Hyperedges: two or more vertices.
Edge cuts approximate communication volume.	Hyperedge cuts accurately measure communication volume.
Assign equal vertex weight while minimizing edge cut weight.	Assign equal vertex weight while minimizing hyperedge cut weight.
	

	ParMETIS Graph			PHG Hypergraph		
Num Procs	Max Nbors	Avg Nbors	Comm Volume	Max Nbors	Avg Nbors	Comm Volume
16	15	12.5	35278	15	11.8	10121
64	53	33.0	56489	40	19.6	22801

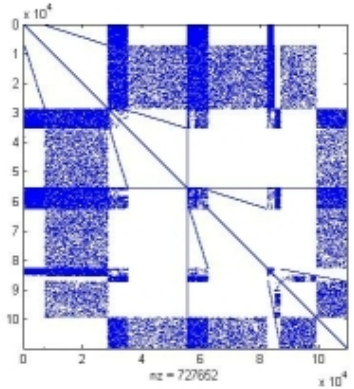
- ~680k Unknown Xyce Circuit Problem (IC)
- 3X Reduction in Communication Volume

Singleton Filter → Block Ordering → Partitioning → Block Jacobi → GMRES

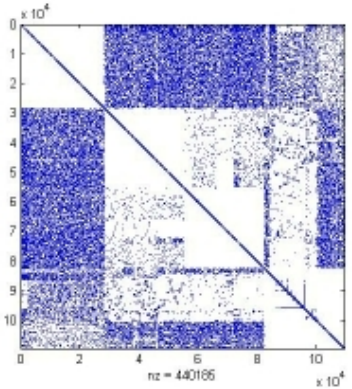
"A Parallel Preconditioning Strategy for Efficient Transistor-Level Circuit Simulation", Thornquist, Keiter, et al, ICCAD 2009

Strategy Comparison: 100K Transistor IC

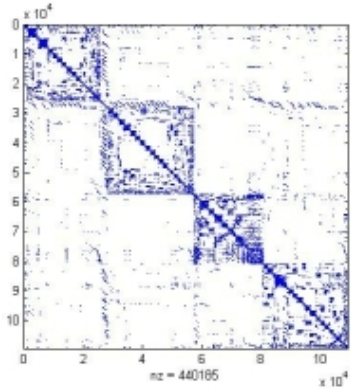
Original



ParMETIS+AMD

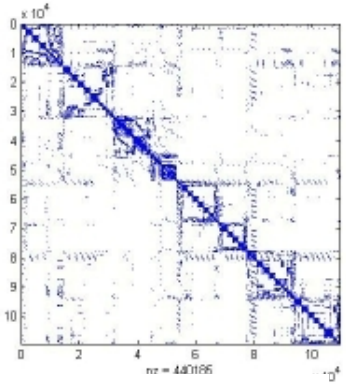


BTF+Hypergraph



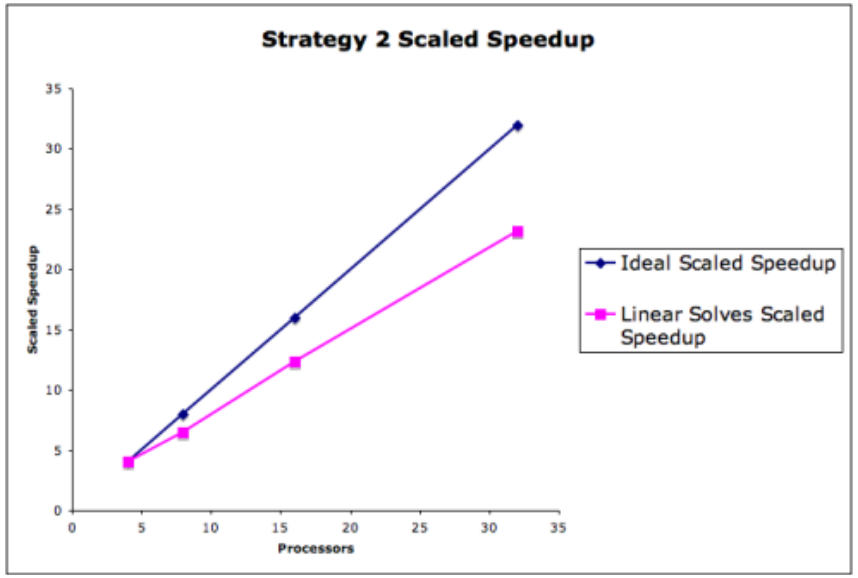
4 processors

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



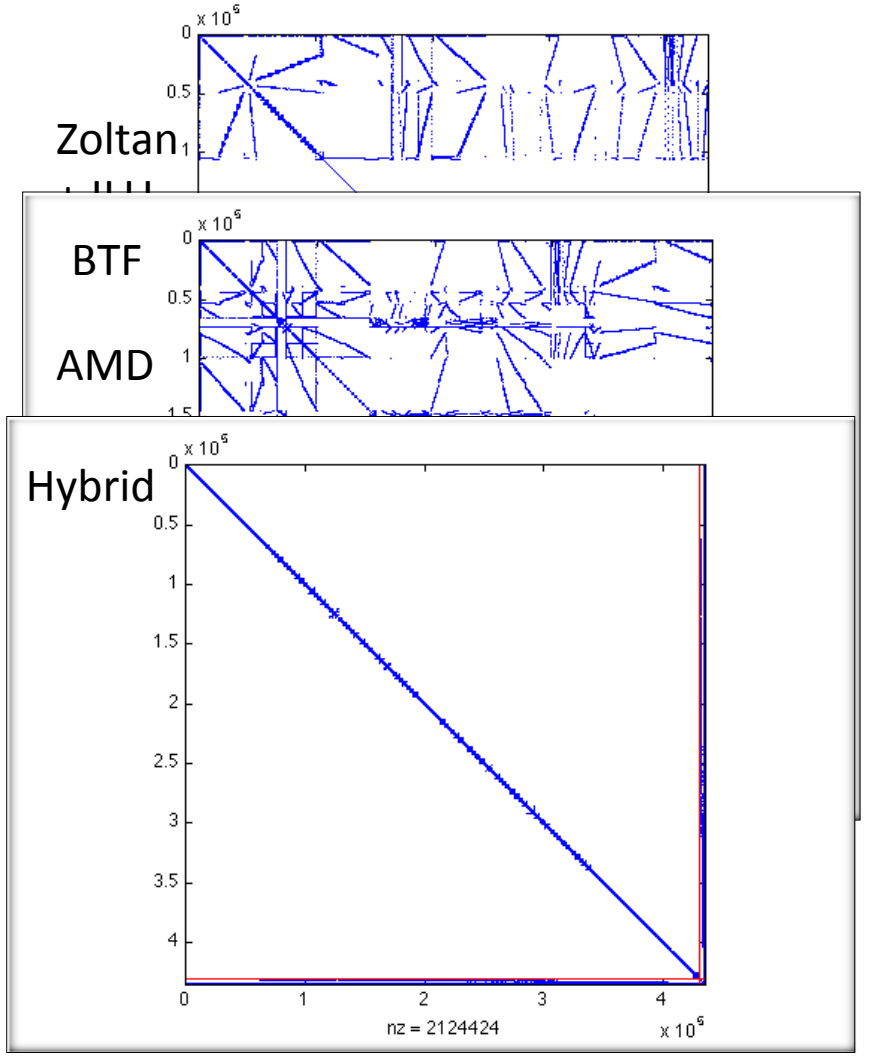
"A Parallel Preconditioning Strategy for Efficient Transistor-Level Circuit Simulation", Thornquist, Keiter, et al, ICCAD 2009

A New Framework for Developing Robust “Hybrid-Hybrid” Linear Solvers

- Sandia ASIC
  - Master tile -> 549144 devices (N=434749)
  - BTF-based preconditioned iterative methods fail
    - BTF: irreducible block size = *334767*

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	



- Led to development of new ”hybrid-hybrid” solver
- Using hybrid partitioning:
  - *Direct : 429974 rows*
  - *Iterative : 4775 rows*

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

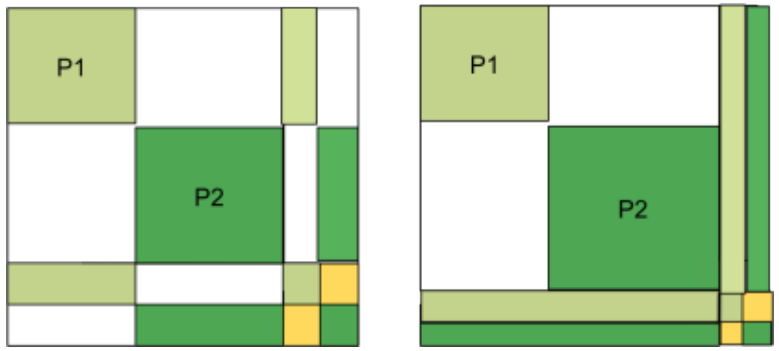
- Solving $Ax = b$ consists of three steps:

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

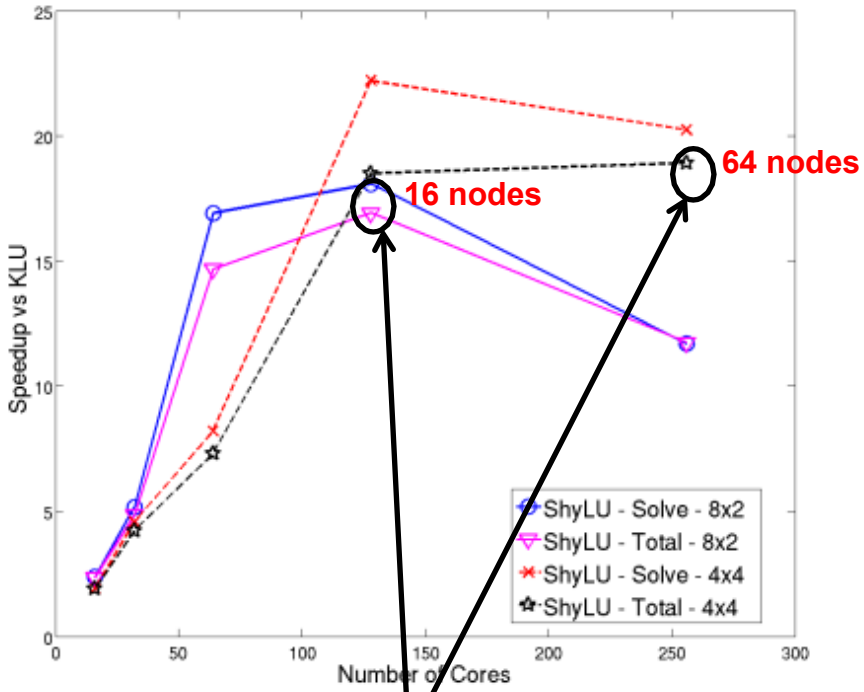


New Linear Solver Achieves 19x Speedup for ASIC Simulation

- 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**, 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)	-	-	-	-	-



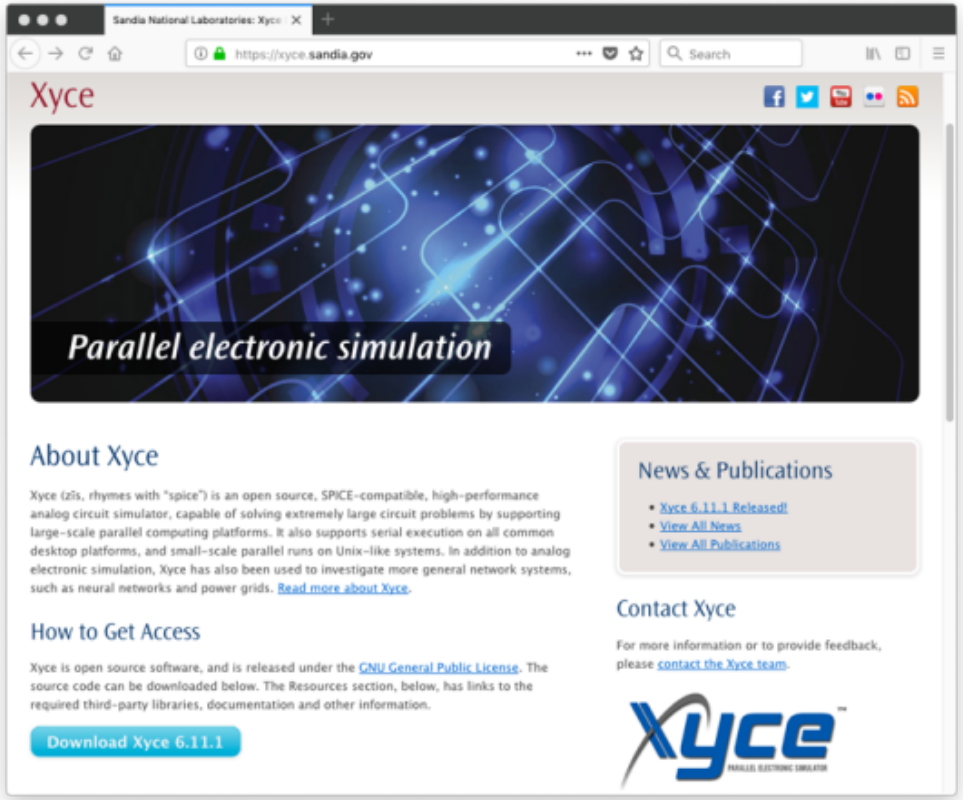
Strong scaling of Xyce's simulation time and ShyLU linear solve time for different configurations of MPI Tasks X Threads per node on TLCC

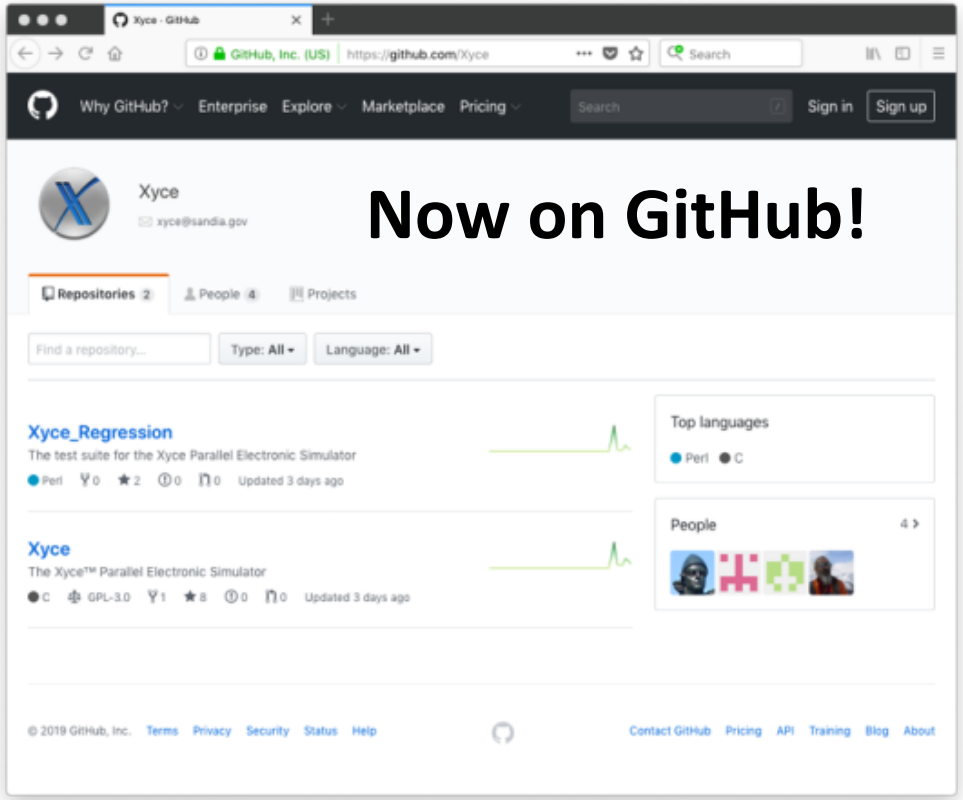
Obtaining Xyce

- Xyce at Sandia
  - **Binary executables** for Windows, OSX and Red Hat Enterprise Linux 6 & 7
  - **Xyce** release source code, **build instructions** and more…
- GitHub
  - For the latest **stable changes** to the **source code** between releases

`https://xyce.sandia.gov`

`https://github.com/xyce`



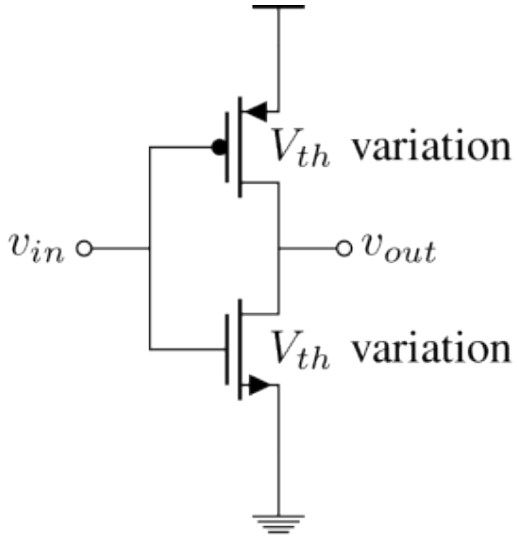


Polynomial Chaos Expansion (PCE) methods in Xyce

Uncertainty Quantification(UQ) in Circuit Simulation

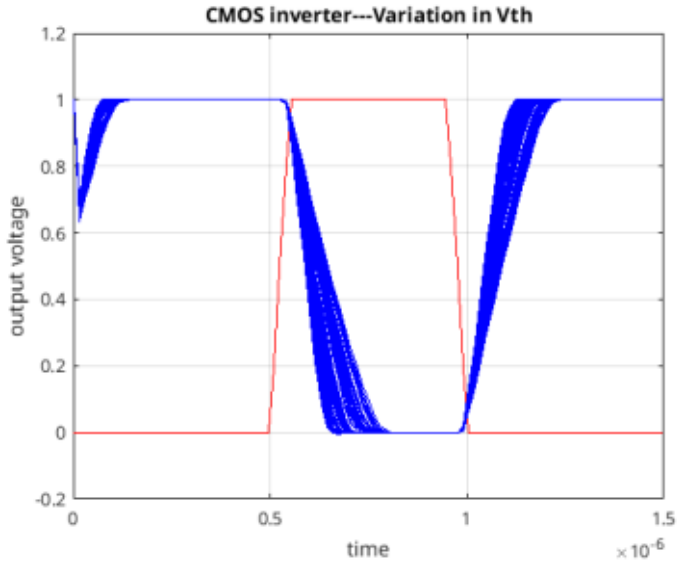
Uncertainty

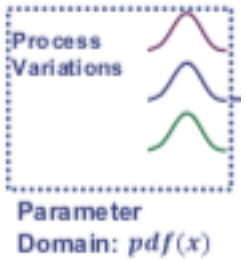
Parameters of devices are subject to statistical variation



Quantification

How much does variation in device parameters affect the output of a circuit?

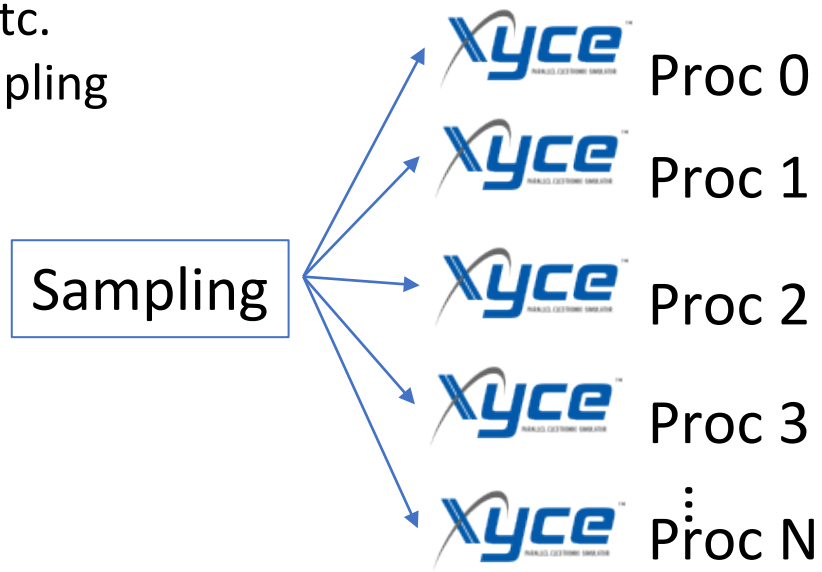




The Obvious Solution: Sampling

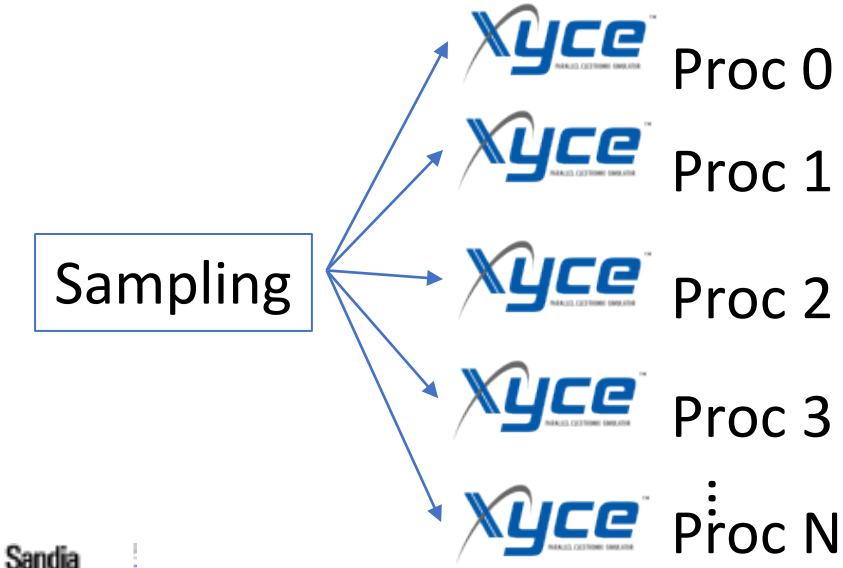
- Sampling Technique
  - Simulate the circuit a large number of times
  - Sample a different set of parameter values for each simulation run
  - Build ensemble averages over the sample outputs
  - Compute statistics of the simulation output: mean, std deviation, etc.
  - Types of sampling: Monte Carlo, Latin Hypercube, importance sampling

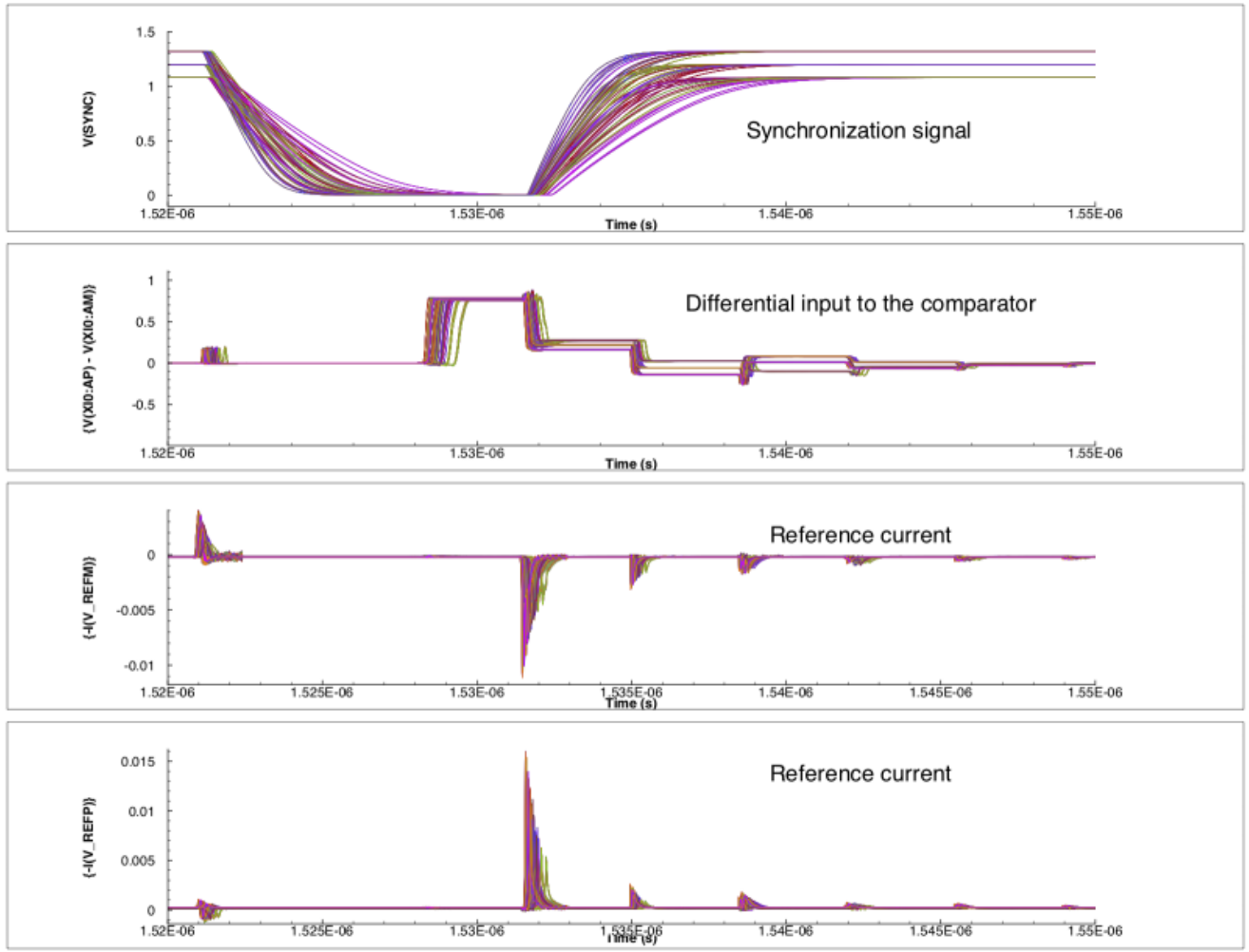
- We would like to avoid it
  - Computationally expensive



Sampling Example: SAR ADC circuit Corner Study (400 corners)

- **SAR ADC** = successive approximation register analog-to-digital converter
- GF65nm 12b SAR ADC by Bindu Madhavan and Edward Lee





Alternative Idea: Non-Sampling Analysis

- Do NOT simulate the circuit multiple times with different parameter samples.
- Instead, run an analysis only once and compute statistics directly from the output.
- Core idea: represent the random system in a way that would enable us to:
  - Convert the stochastic system into a different system with no random components.
  - Represent the statistics of the stochastic system in terms of the output of this new system

Polynomial Chaos Expansion (PCE)

- Do NOT simulate the circuit thousands of times a la sampling
- Instead, run a single analysis and compute statistics directly from the output
- Method: Polynomial Chaos Expansion (PCE)

- Represent uncertain quantities as a finite sum:

$$O(p) \approx \hat{O}(p) = \sum_{i=0}^{P} \alpha^i \Psi^i(p)$$

Numerical Methods for Stochastic Computations: A Spectral Method Approach, D. Xiu, Princeton University Press

- O is a scalar objective function
- p is a vector of uncertain parameters
- Ψ is a set of orthogonal polynomials, such as Hermite polynomials, Legendre, etc
- α are the scalar coefficients
- Truncating the PCE is justified because of the Cameron-Martin theorem

Different polynomials for different distributions

	Distribution of Z	gPC Basis Polynomials
Continuous	Gaussian	Hermite
	Gamma	Laguerre
	Beta	Jacobi
	Uniform	Legendre
Discrete	Poisson	Charlier
	Binomial	Krawtchouk
	Negative binomial	Meixner
	Hypergeometric	Hahn

PCE can be Intrusive, non Intrusive or in-between

Non Intrusive

Kind of like sampling in that an outer loop is applied to existing simulator.

post-processing to determine PCE coefficients

Implementation is easy, as simulator doesn't require much (any?) modification.

Subject to interpolation error and integration error

Examples:
Stochastic Collocation (SC)
Non Intrusive Spectral Projection (NISP)
Regression methods

Partially Intrusive

Very similar to non-intrusive, but the loop is applied *inside the solver*.

It is an inner loop, rather than an outer loop. Can exploit this for parallelism, etc. Otherwise, however the underlying differential equations are the same, and each sample point is still de-coupled.

Examples:
Stochastic Testing (ST)

Fully Intrusive

The polynomial expansion is applied to every term of the underlying DAE system.

This results in a new (block) system of equations. The blocks are strongly coupled, so efficient linear solves are a challenge.

Implementation is challenging.

Optimal accuracy – residue is orthogonal to the linear space spanned by gPC polynomials

Examples:
Stochastic Galerkin Method

In Xyce we've implemented the whole continuum of methods.

Types of PCE

- We studied 3 types of PCE in this project
  - Non-intrusive Spectral Projection (NISP)
    - evaluates the coefficients α using Galerkin projection:

$$\alpha^i = \frac{\langle O\Psi^i \rangle}{\langle (\Psi^i)^2 \rangle} = \frac{1}{\langle (\Psi^i)^2 \rangle} \int_\Gamma O(p;y)\Psi^i(y)\rho(y)dy = 0, \quad i = 0,\ldots,P.$$

  - Non-intrusive Regression PCE
    - evaluates the coefficients α using Least Squares Regression
  - Fully intrusive Spectral Projection
    - propagates the PCE expansion all the way thru the DAE system of equations, and solves directly for coefficients α for every solution variable.

- Sometimes Spectral Projection methods are also referred to as quadrature methods
- The first two were implemented in a “partially intrusive” manner; they compliment embedded sampling, and can be computed at every time step to get time-dependent uncertainties.
- To the best of our knowledge, Xyce is the only circuit simulator (free or commercial) to have implemented these methods.

Fully Intrusive PCE

- Fully intrusive PCE requires the largest amount of code modification.
- The Gakerkin project equation is applied to all of the terms of the DAE:
- So, the system of equations uses new block vectors:

$$x(p) \approx \hat{x}(p) \equiv \sum_{i=0}^{P} x^i \Psi^i(p).$$

$$f(p) \approx \hat{f}(p) \equiv \sum_{i=0}^{P} f^i \Psi^i(p).$$

$$X = \begin{bmatrix} x^0 \\ \vdots \\ x^P \end{bmatrix} = \sum_{k=0}^{P} e^k \otimes x^k, \quad F = \begin{bmatrix} f^0 \\ \vdots \\ f^P \end{bmatrix} = \sum_{k=0}^{P} e^k \otimes f^k$$

- And now, the nonlinear system solved via Newton's method is F(X) = 0
- This requires that a Stochastic block Jacobian matrix ∂F/∂X be constructed via:

$$\frac{\partial f^i}{\partial x^j} = \frac{1}{\langle (\Psi^i)^2 \rangle} \int_\Gamma \frac{\partial f}{\partial x}(\hat{x}(y); y) \Psi^i(y) \Psi^j(y) \rho(y) dy \approx \sum_{k=0}^{P} A^k \frac{\langle \Psi^i \Psi^j \Psi^k \rangle}{\langle (\Psi^i)^2 \rangle}$$

where

$$\frac{\partial f}{\partial x}(\hat{x}(p); p) \approx \sum_{k=0}^{P} A^k \Psi^k(p)$$

$$A^k = \frac{1}{\langle (\Psi^i)^2 \rangle} \int_\Gamma \frac{\partial f}{\partial x}(\hat{x}(y); y) \Psi^k(y) \rho(y) dy, \; k = 0, \ldots, P$$

Voltage limiting for Fully Intrusive PCE

- Circuit simulation relies on a unique nonlinear solver enhancement called *voltage limiting*
  - Without it, circuit simulation is not viable. Other nonlinear solution methods don't work
- This algorithm is not well-described in the literature, until our recent paper on the topic*.
- Furthermore, it is especially not well described for exotic algorithms such as PCE.

$$\frac{\partial F}{\partial X}\Delta X_l = -F(X_l) + \frac{\partial F}{\partial X}\Delta X_l^{lim} = -F(X_l) + F^{lim}$$

$$F^{lim} = \frac{\partial F}{\partial X}\Delta X^{lim},$$

$$F^{lim} = \begin{bmatrix} f^0 \\ \vdots \\ f^P \end{bmatrix} = \sum_{k=0}^{P} e^k \otimes f_{lim}^k$$

- (For details, see the final SAND report)
- PCE voltage limiting has not been previously published.

* K. V. Aadithya, Eric Keiter, and Ting Mei, "Predictor/Corrector Newton-Raphson (PCNR): A Simple, Flexible, Scalable, Modular, and Consistent Replacement for Limiting in Circuit Simulation", to appear in *Proceedings of 12th International Conference On Scientific Computing In Electrical Engineering*

Intrusive PCE Linear System

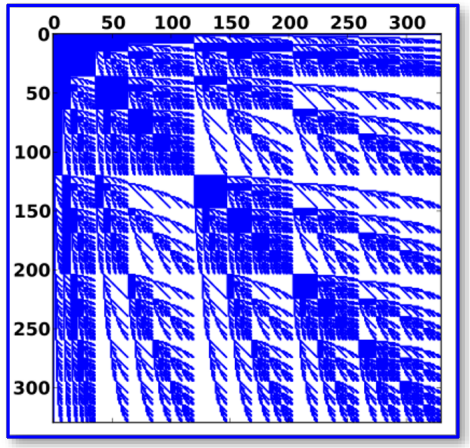
Circuit DAE equation

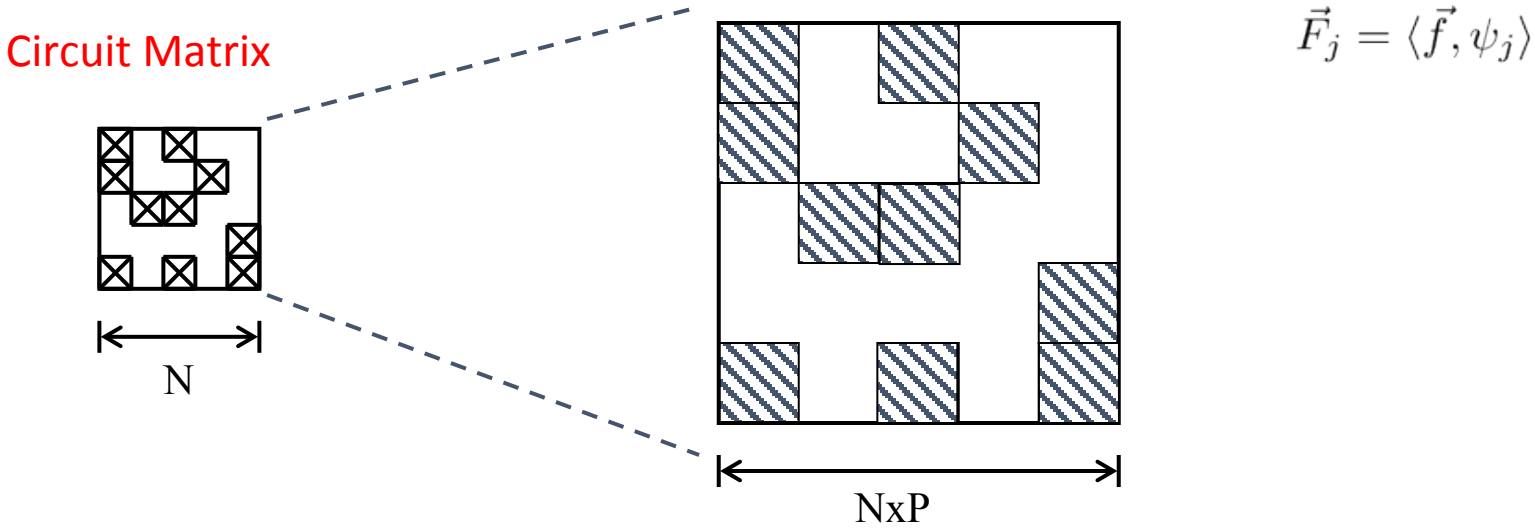
$$\frac{\mathrm{d}}{\mathrm{d}t}\vec{q}(\vec{x}(t),\vec{\varphi})+\vec{f}(\vec{x}(t),t,\vec{\varphi})=0$$

PCE DAE equation

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\vec{Q}_1(\vec{u}(t))\\ \vdots \\ \vec{Q}_P(\vec{u}(t))\end{bmatrix}+\begin{bmatrix}\vec{F}_1(\vec{u}(t),t)\\ \vdots \\ \vec{F}_P(\vec{u}(t),t)\end{bmatrix}=0$$

$$\vec{Q}_j=\langle\vec{q},\psi_j\rangle$$
$$\vec{F}_j=\langle\vec{f},\psi_j\rangle$$





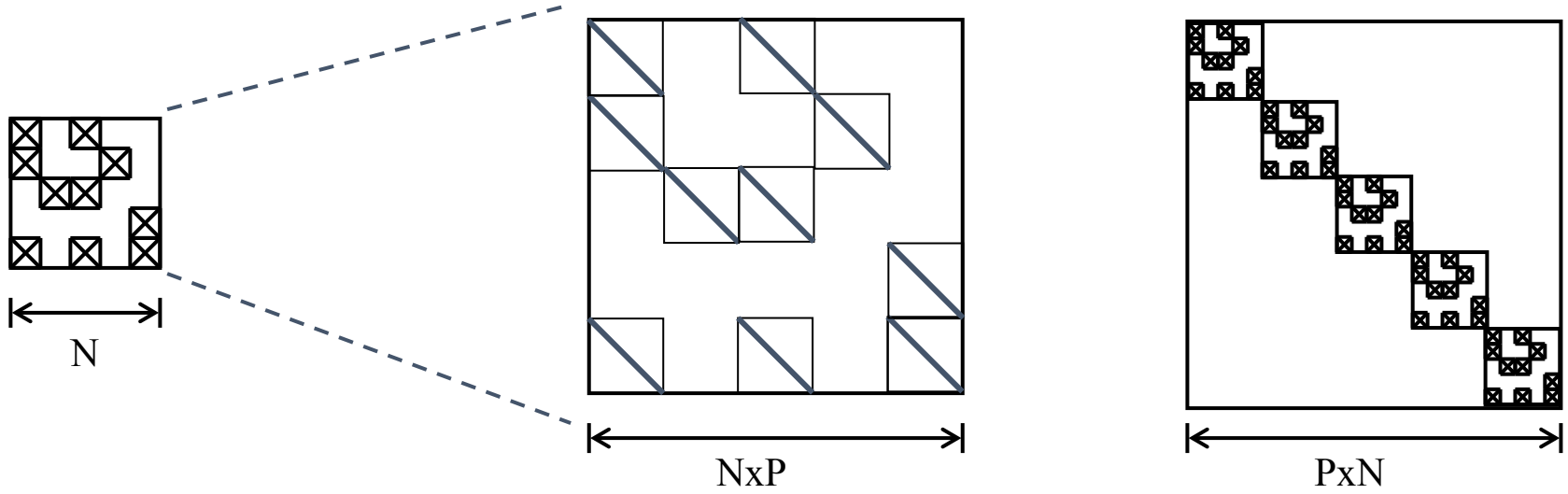
N = circuit unknowns
P = sample or quad points

- Block linear solvers (BASKER), next gen platforms, Stokhos/Kokkos libraries

Embedded Linear System

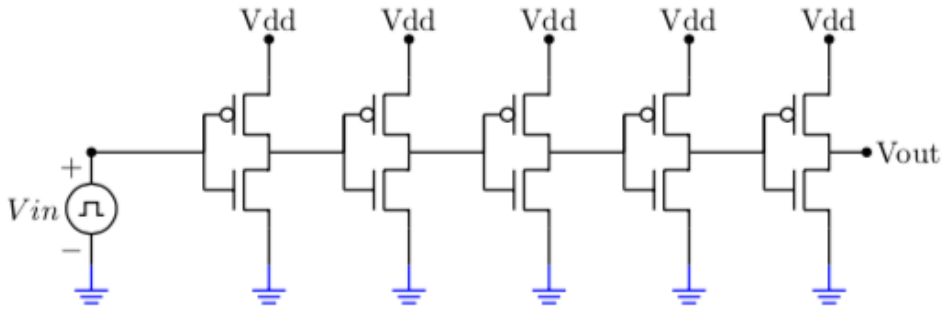
Block structure options:

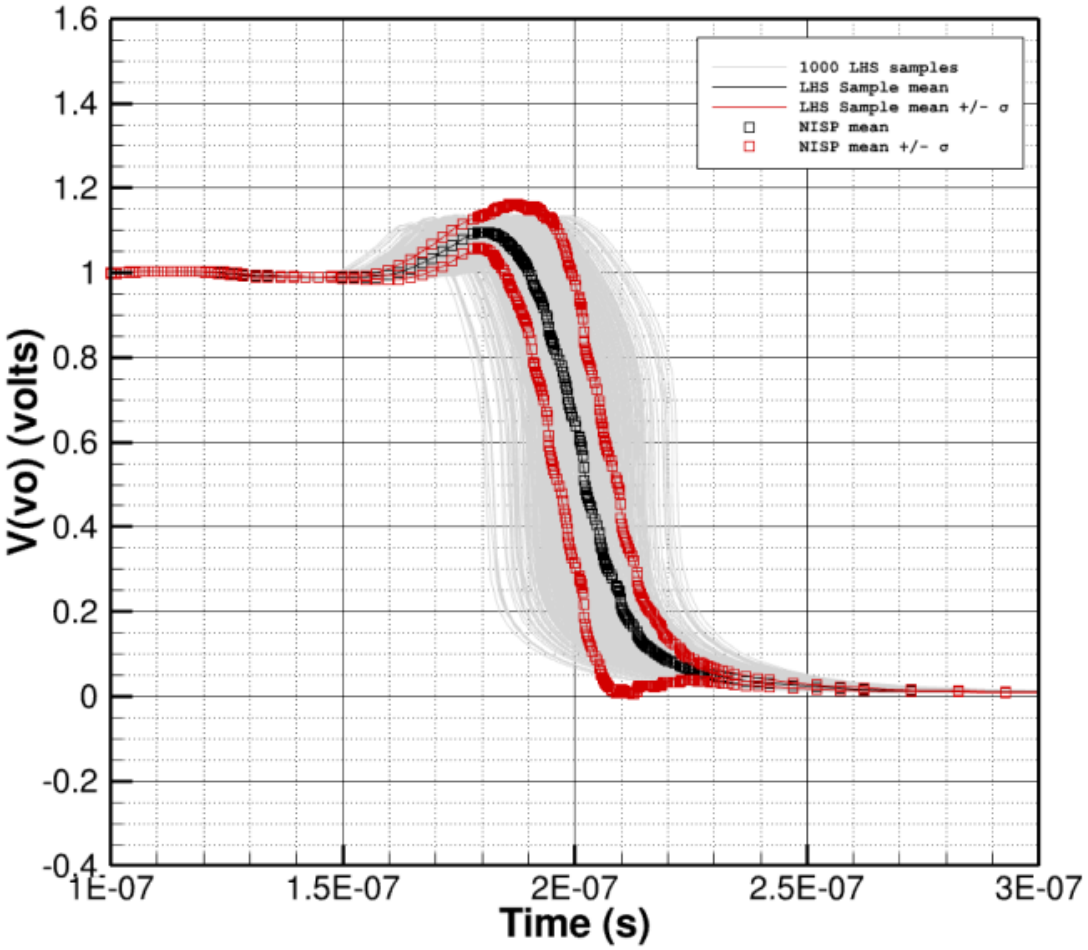
1. Expand each entry of original DAE Jacobian to be a P-sized block (left)
2. Have diagonal matrix of blocks; each block is original sparse DAE matrix (right)



- N… circuit unknowns (x)
- P… Number of samples. (in this example, P=5
- For a purely serial implementation, option 2 (right) is faster; ~linear scaling with P
- For block solvers (BASKER) option 1 (left) faster
- Implemented both structures in Xyce

Simple example: CMOS inverter.



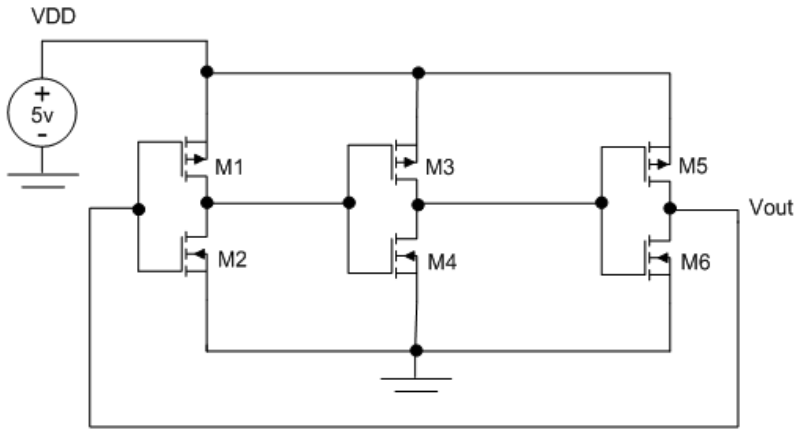


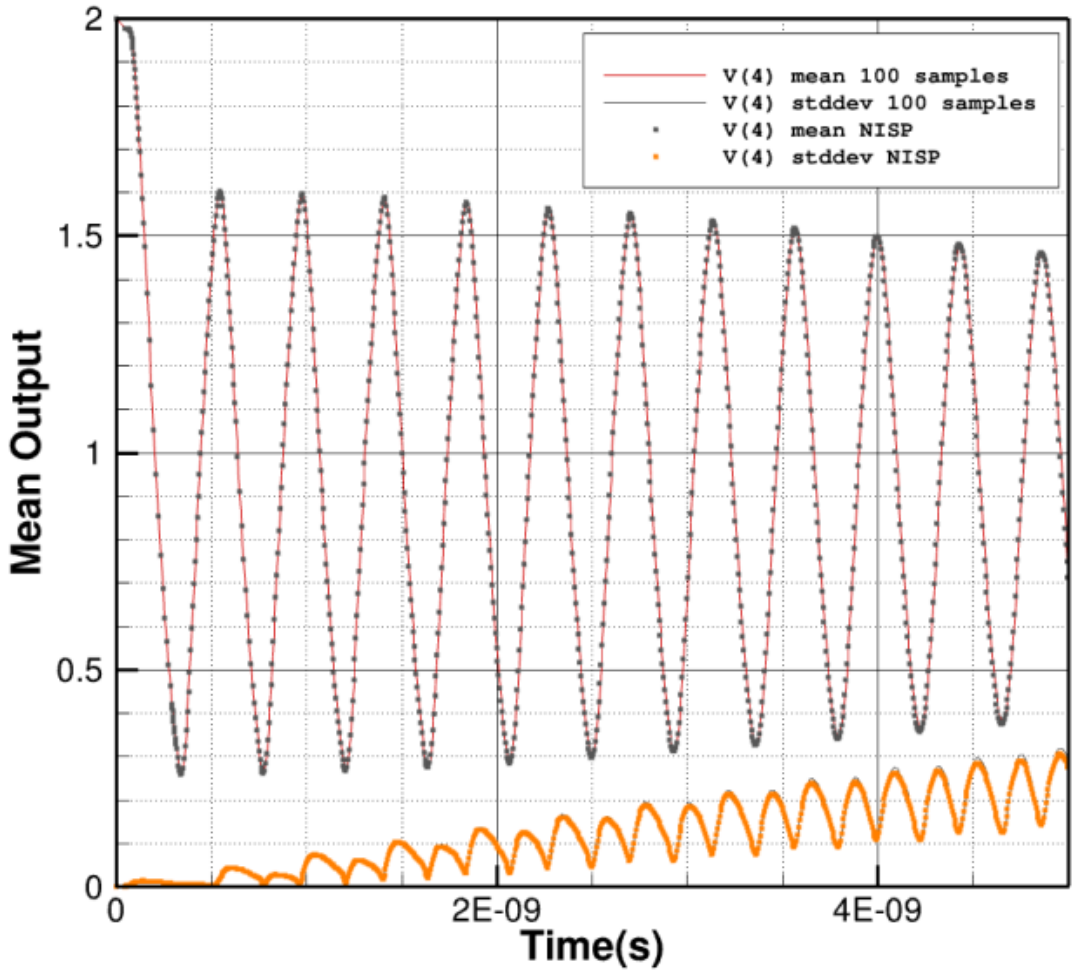
- Number of parameters with variability (M): 2
- Highest polynomial order: 4
- Total number of quadrature points: 25

- Comparison is between LHS sampling and the “semi-instrusive” form of quadrature PCE, also known as NISP.

- Two uncertain parameters, threshold voltage on both MOS devices.

Simple example: 3-stage Ring Oscillator



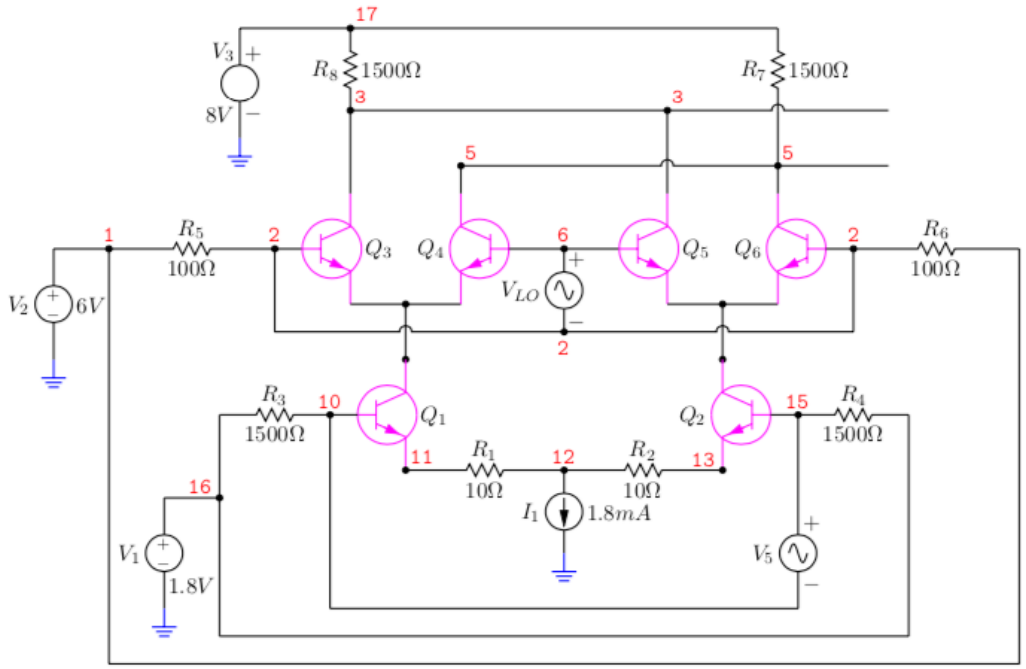


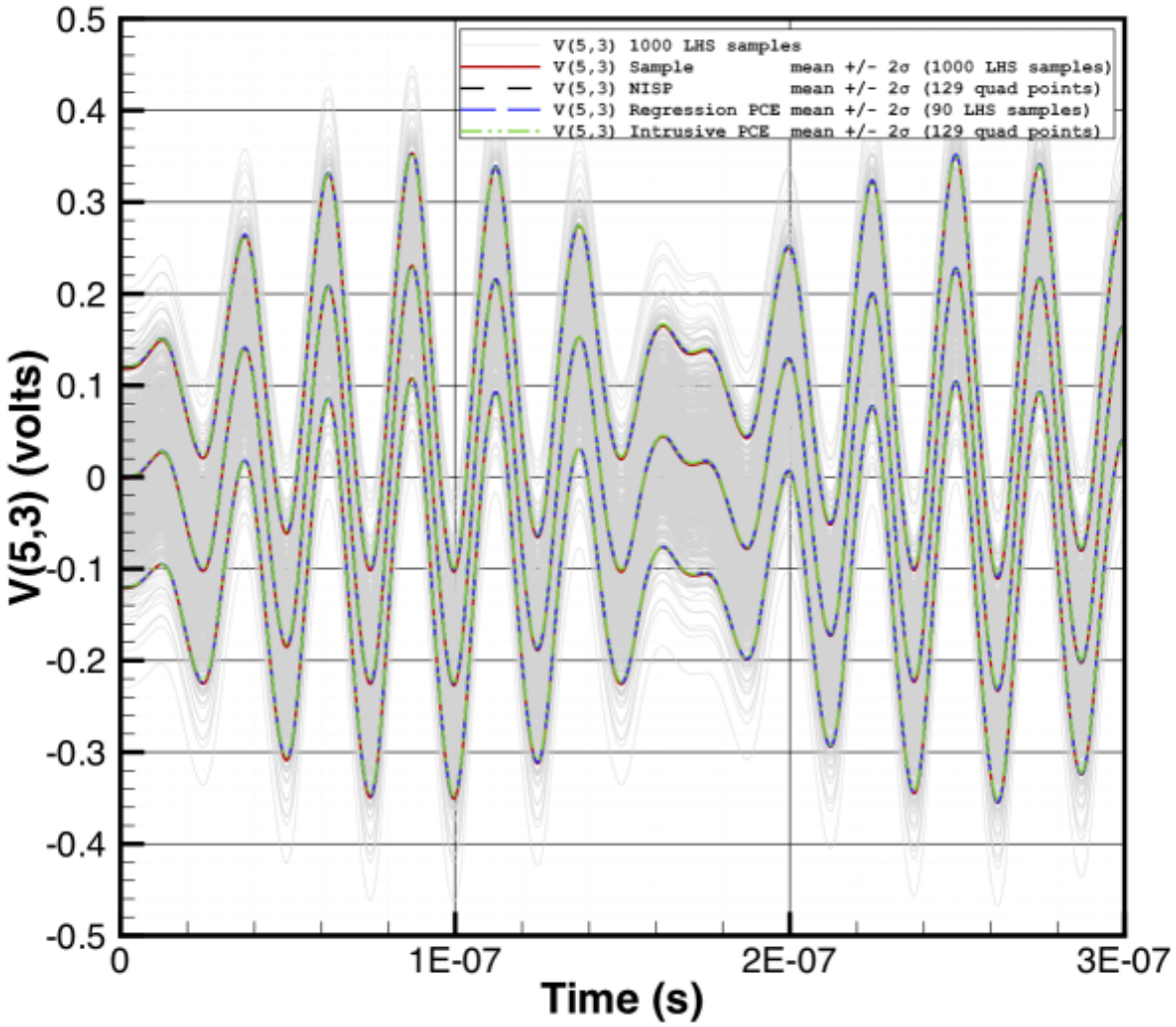
- Number of parameters with variability (M): 1
- Highest polynomial order: 5
- Total number of quadrature points: 6

- Comparison is between LHS sampling and the “semi-instrusive” form of quadrature PCE, also known as NISP.

- Single uncertain parameter, threshold voltage on one of the MOS devices.

Gilbert Cell Mixer PCE results





Ex. Input file commands

```
.embeddedsampling
+ means              = 10,10,1500,1500,100,100,1500,1500
+ std_deviations =  1, 1,  50,  50,  5,  5,  50,  50
+ param =     R1:R,R2:R,R3:R,R4:R,R5:R,R6:R,R7:R,R8:R
+ type = normal,normal,normal,normal,
+        normal,normal,normal,normal

  .options embeddedsamples projection_pce=true
+ outputs={v(5,3)} order=2 sparse_grid=true
```

Method	Number of Samples	Runtime (sec)	Scaled runtime
LHS	1000	54.9	1152x
NISP	129	6.02	126x
Regression PCE	90	4.5	94x
Intrusive PCE	129	85.74	1799x

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:

https://github.com/xyce
https://xyce.sandia.gov
xyce@sandia.gov

Google Group Forum:

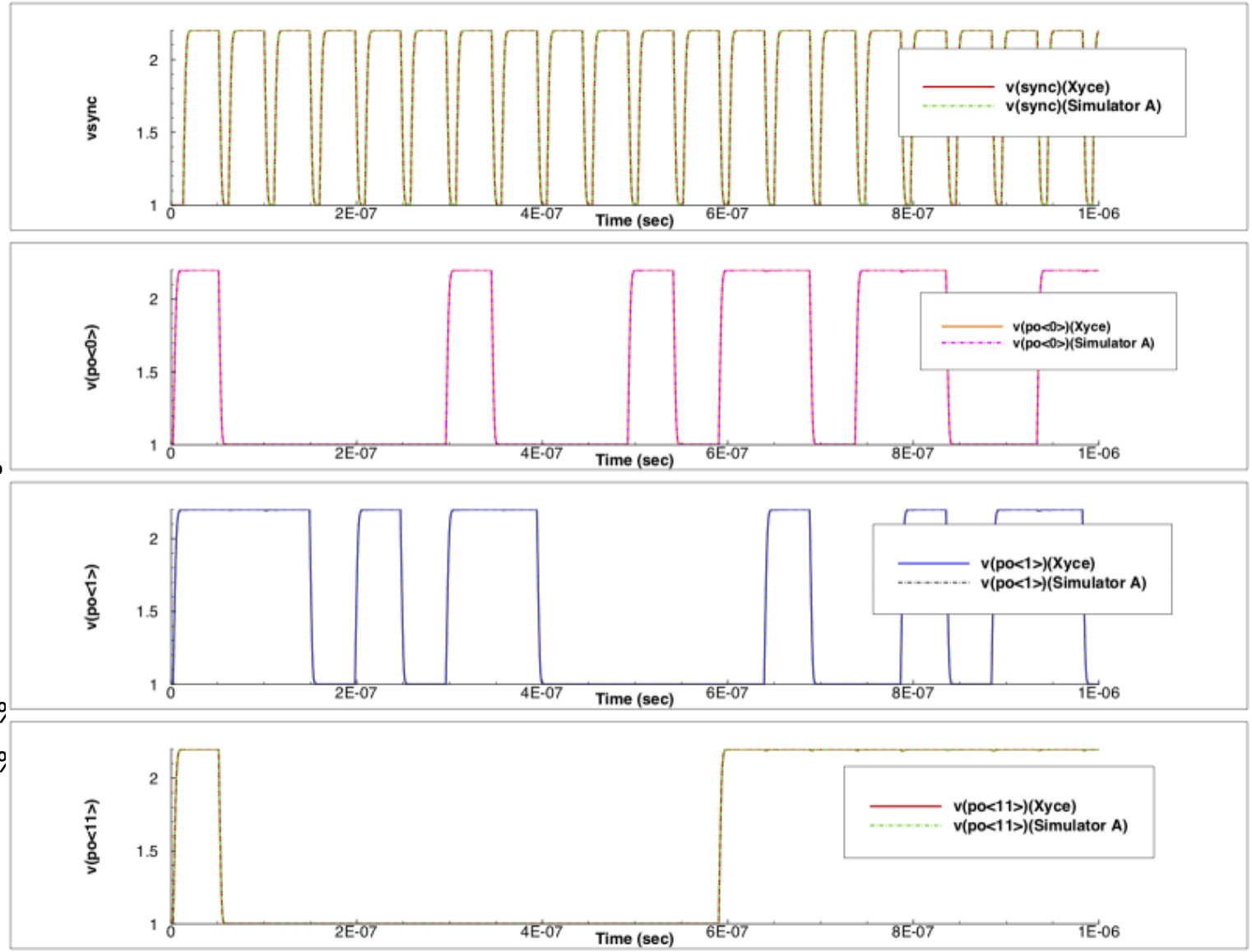
https://groups.google.com/group/xyce-users

Extra slides

Comparison of SAR ADC Result, Xyce vs Commercial Simulator

- **Results match well. RMS Errors small**

- RMS relative error in v(sync) is 0.0434905680015374%
- RMS relative error in v(po<0>) is 0.0232474456164593%
- RMS relative error in v(po<1>) is 0.023581461963474%
- RMS relative error in v(po<2>) is 0.02583082511786%
- RMS relative error in v(po<3>) is 0.0240096727254828%
- RMS relative error in v(po<4>) is 0.0166525520072121%
- RMS relative error in v(po<5>) is 0.00929693070847055%
- RMS relative error in v(po<6>) is 0.0309201017241085%
- RMS relative error in v(po<7>) is 0.0230237794341722%
- RMS relative error in v(po<8>) is 0.0259005260949305%
- RMS relative error in v(po<9>) is 0.0175662606806119%
- RMS relative error in v(po<10>) is 0.00940986678122403%
- RMS relative error in v(po<11>) is 0.00976999004888706%



SAR ADC timings, Xyce vs commercial simulators

- Recent efficiency improvements to Xyce have brought it close to “Simulator B” for one processor.
- Still work to do to catch “Simulator A”.
- Some of the difference is due to BYPASS, which is present in ”Simulator A”, but not Xyce or “Simulator B”.