Parallel Circuit Simulation: How Good Can It Get?

Andrei Vladimirescu

isep

Overview

- Opportunities for Full-Chip Analog Verification
- Analog vs. Digital Design
- SPICE – standard design tool for Analog and Mixed-Signal
- Functionality, Robustness and Performance
- State-of-the-art Parallel HW – Potential for increased SPICE productivity
- FastSPICE
- Results

Speedup: Opportunities and Options

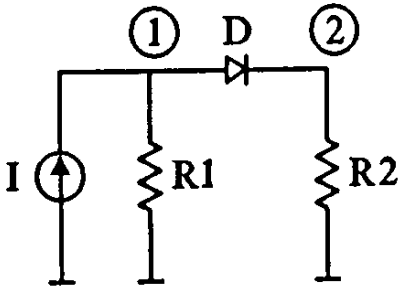
- Opportunities
  - Multi/Many-core (MPU) - MIMD Architecture
  - Graphic Porcessors (GPU) – SIMD Architecture
- Options
  - FastSPICE
  - Good old SPICE
    - Same solution algorithms tailored for parallel computation
  - ?

Analog vs. Digital Circuit Design

- All ICs are built w/ transistors, but
  - Analog ICs are characterized in Volts and Amperes
  - Digital ICs can be abstracted in « 1 » and « 0 »
  - Design Approach differs
- Main CAD Approach
  - Analog: Designer creates circuit, SPICE validates
  - Digital: Logic Synthesis Tool generates circuit based on Standard Cell Libraries
  - Mixed-Signal: SPICE-Verilog (AMS)?
    - Is it still appropriate at V_{DD} = 0.6V?

The Problem (Analog) SPICE Solves

- Numerical Representation of Circuit Topology



Nodal Admittance Matrix
Setup from Incidence Matrix

- Circuit Equations

BCE $V_1 = R_1 I_{R1}$ or $I_{R1} = G_1 V_1$

$$I_D = I_S \left(e^{\frac{V_{12}}{V_T}} - 1\right) \approx G_D V_{12} - I_{DN}$$

KCL $(G_1 + G_D) V_1 - G_D V_2 = I + I_{DN}$

$-G_D V_1 + (G_2 + G_D) V_2 = -I_{DN}$

→ Needs Nonlinear Device Model

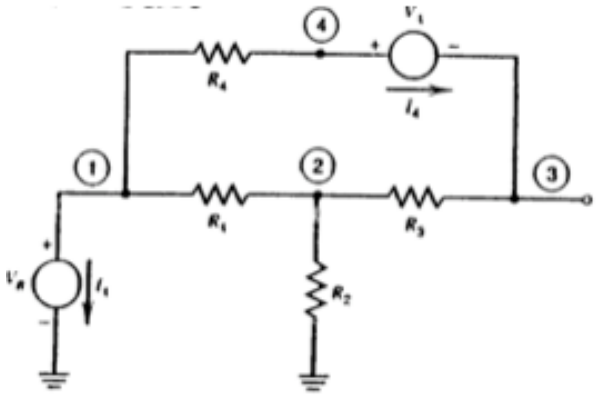
- Linear Equation Solution → Sparsity

$$Y \bullet v = i$$

→ Needs Fast, Accurate Solver

Electric Circuit Representation

$$
\begin{array}{lccccccc}
(1) & (G_1+G_4)V_1 & -G_1V_2 & & & -G_4V_4 & +I_1 & = 0 \\
(2) & -G_1V_1 & +(G_1+G_2+G_3)V_2 & -G_3V_3 & & & & = 0 \\
(3) & & -G_3V_2 & +G_3V_3 & & & -I_4 & = 0 \\
(4) & -G_4V_1 & & & & +G_4V_4 & +I_4 & = 0 \\
(V_B) & V_1 & & & & & & = V_B \\
(V_A) & & & -V_3 & +V_4 & & & = V_A
\end{array}
$$



$$
\left|\begin{array}{cccc|cc}
G_1+G_4 & -G_1 & 0 & -G_4 & 1 & 0 \\
-G_1 & G_1+G_2+G_3 & -G_3 & 0 & 0 & 0 \\
0 & -G_3 & G_3 & 0 & 0 & -1 \\
-G_4 & 0 & 0 & G_4 & 0 & 1 \\
\hline
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 1 & 0 & 0
\end{array}\right|
\left\|\begin{array}{c} V_1 \\ V_2 \\ V_3 \\ V_4 \\ \hline I_1 \\ I_4 \end{array}\right\|
=
\left\|\begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \\ \hline V_B \\ V_A \end{array}\right\|
$$

Modified Nodal Analysis

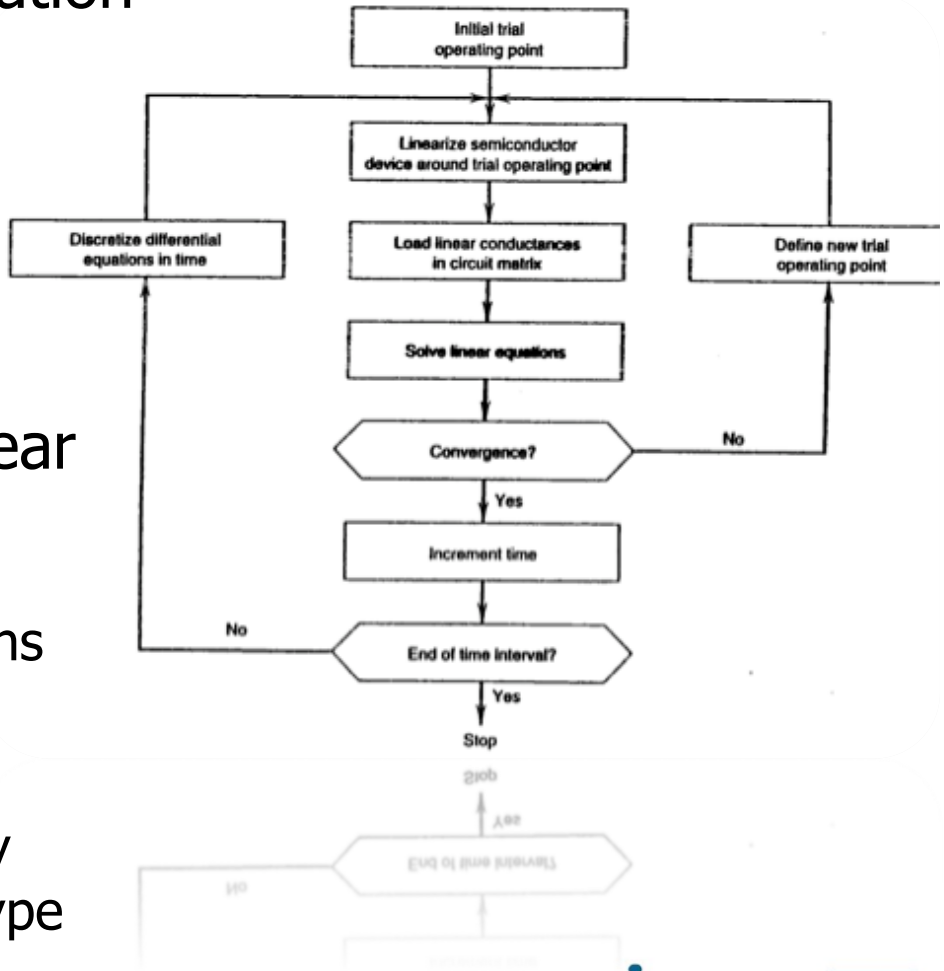
Current src

$$
\begin{bmatrix} G & F \\ B & R \end{bmatrix}
\begin{bmatrix} V \\ I \end{bmatrix}
-
\begin{bmatrix} C \\ E \end{bmatrix}
$$

[1] Vladimirescu, A., *The SPICE Book*, J. Wiley, NY, 1994

SPICE Solution: Two Parts

- Nonlinear Device Model Evaluation
  - Partial derivatives dI/dV
  (Y matrix entries)
  - Floating-point intensive
  - Matrix-load bottleneck
  - Increases linearly w/ # devices
- Nonlinear ODEs -> Sparse linear equation solution
  - Memory intensive
  - Increases superlinearly w/ # eqns
- Ratio between the two
  - determines speed vs. complexity
  - Depends on circuit and model type



Current IC Design Approach

- Process-corner based analysis
  - Best and worst-case NMOS and PMOS
    - Typical, FF, SS, FS and SF
- Monte-Carlo statistical simulation
  - Global and local variations of key parameters
- Reduced-sample Monte Carlo
  - Covers extremities of distribution
- Independent simulations
  - Speedup/Productivity gain is Nx for N processors

Feasibility of Full-chip Simulation

- How about million-transistor circuit SPICE run?
- Algorithms proven to work for upwards of 10^6 transistors
  - Convergence (nonlinear eqns.) is key issue
  - Time step selection (differential eqns) – key for performance
- Issue for feasibility: Speed @ same Accuracy
  - Parallelization should cut run time, but
  - By how much?

State-of-the-Art SPICE Work

- New Functionality
  - Algorithms
  - Semiconductor device models
  - Model development language (Verilog-AMS)
  - Add-on Tools for Stats/Yield Optimization (Solido, MunEDA)
- Robustness beyond 100,000 transistors
  - Automatic exercise of continuation methods (homotopy)
  - 100,000 transistors and beyond
- Faster simulation/throughput
  - Parallel Direct-Methods: Spectre/APS/XPS, HSPICE/XA, AFS, Eldo
  - Fast-SPICE: CustomSim (HSim, NanoSim, FineSim), UltraSim
  - Public Domain Research: NGSPICE, Xyce

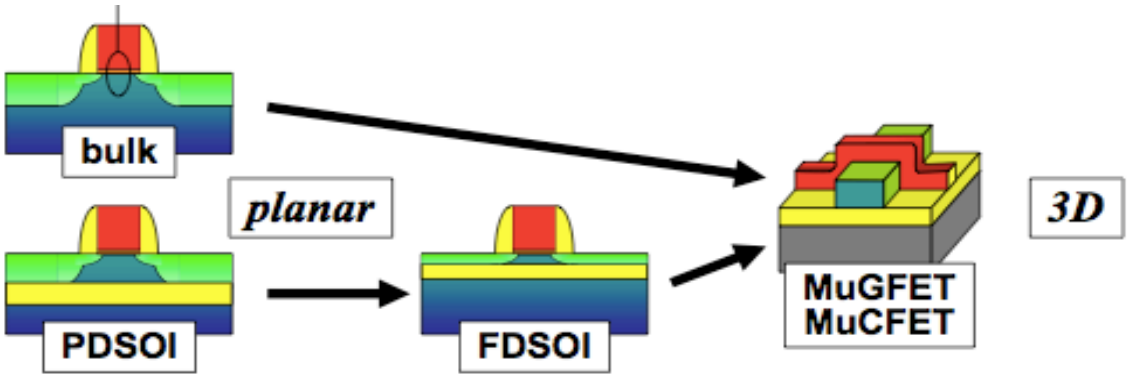
Algorithms

- Large-signal time/frequency-domain analysis
  - Harmonic Balance
  - Steady-State Analysis
  - Envelope Simulation
  - RC reduction
  - Linear/Nonlinear Matrix Partitioning
  - S-Parameter Simulation for Chip/Package/PCB
- Continuation Methods (Homotopy)
- Improved Numerical Integration/Time-Step Control
- Parallel and Fast-SPICE Techniques

➢ Above lead to Increased Simulation Speed and Capacity

Semiconductor Models – MOSFET Today

- Models reflect today's device structures
  - Fourth Generation BSIM4 and PSP
    - 65, 45, 28nm technologies
  - Emerging BSIM6 and BSIM-IMG/CMG
    - 28, 22, 20/14 nm bulk, FDSOI and FinFET



State-of-the-Art Compute HW

- Single-Instruction Mutiple-Data (SIMD)
  - Cray
  - Nvidia GPU is multi-SIMD processor
    - 128/256 SIMD CPUs, 12k threads, 50-200 Gflops (SP)
- Multiple-Instruction Multiple-Data (MIMD)
  - Multi-core: 4-8 CPUs on a chip
  - Many CPUs in a cloud

First Parallel SPICE in 1980
CLASSIE Project

- Today's complex ICs and Multi/Many-Core processors
  - Faster simulation same accuracy – use parallelism
- Main ideas from CLASSIE[1] on an SIMD architecture (Cray-1) have been implemented today
  - Good progress in commercial simulators
    - Hspice, Spectre, Eldo, FineSim, AFS
- Two main components of the simulator execution time:
  - Model evaluation - easy
  - Linear equation solution (sparse system) - hard
- Large circuits get most advantage
  - Decoupling blocks, BBDF matrix
  - Still left with synchronizing the border – serial
- SPICE is both floating-point and memory intensive

[1] Vladimirescu, A., *LSI Circuit Simulation on Vector Computers,* ERL Memo UCB/ERL M82/75, Univ. of California, Berkeley, 1982

Device Evaluation: Easy

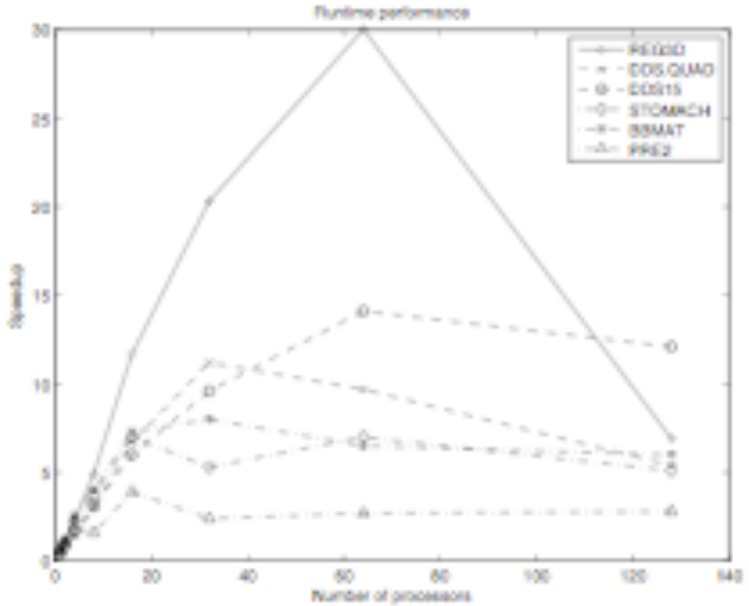
- Two parts of a transistor in SPICE
  - Instance in the circuit – 1,000s to 100,000s
  - Model parameters (coefficients of device eqns.) – few to 10s
- Partitioning/Parallelization options
  - SIMD
    - Setup long vectors of all devices using same model
    - For each device compute all possible values of G (dI/dV)
    - Retain the G value corresponding to the operating region
  - MIMD
    - Divide the devices roughly equal to processors (by model)
    - Compute Gs in parallel

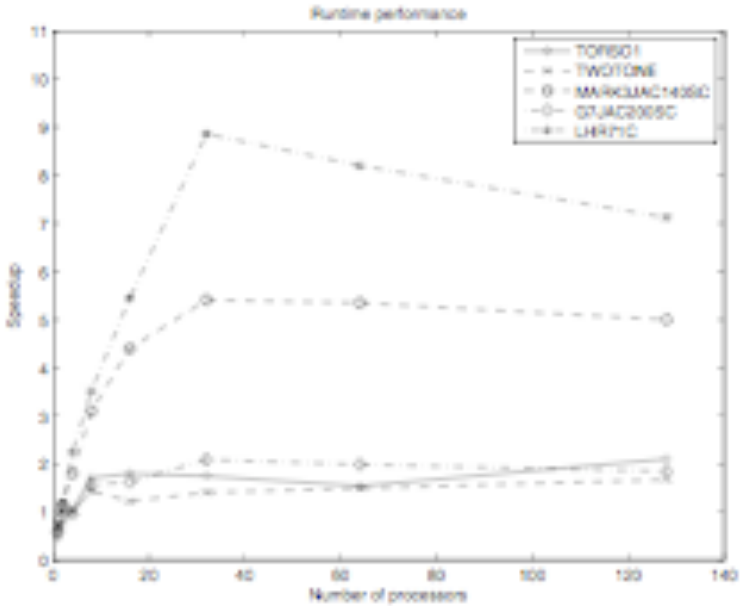
Sparse Equation Solution: Hard

- Sparse matrix entries saved as vector
  - 3 vectors of indices for matrix location
  - 3 memory accesses for retrieving each matrix entry
  - LU factorization and back substitution are sequential
  - Superlinear time increase w/ circuit size
- Approaches to Speedup/Parallelization
  - Generation of matrix-specific loopless machine code
  - Graph Partitioning
  - Hierarchical Matrix Partitioning
    - Based on Circuit Hierarchy
    - Tearing

Graph-Partitioning Parallelization

- Alternate approach to Sparse-Matrix solution
  - Symbolic factorization parallelization
  - Sparsity preservation
- Partitioning -> separator tree -> dependency-free computations

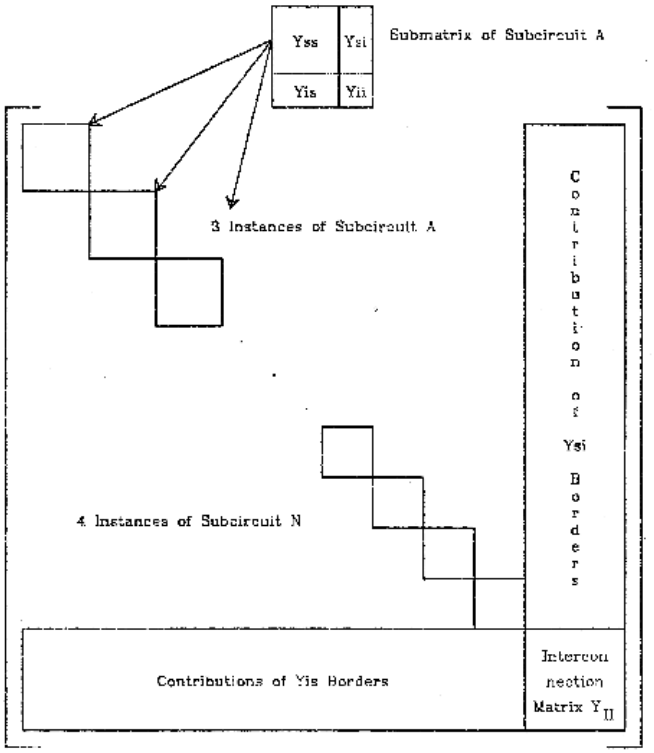




L.Grigori, J.Demmel, X.S.Li, Parallel Symbolic Factorization for Sparse LU with Static Pivoting, *SIAM JSC,* Vol 29, No. 3, 2007

Hierarchy Parallelization

- Large circuits
  - formed of small number of blocks
- Each circuit block repeated
  - Set up matrix to reflect -> BBDF
  - Each block type
    - Solved in vector mode for all occurrences
    - MIMD: allocated to one processor and occurances multi-threaded
- Issue: Border of BBDF
  - Solved sequentially



Tearing Parallelization

- Alternate automatic approach to partition circuit matrix
  - Automatically detect weakly coupled blocks
  - Allocate each new block to a processor (MIMD)
- Solve by two-level Newton
  - First level: solve each block
  - Second level: reconcile solution at circuit level
  - Additional iterations and overhead

FastSPICE - Common Features

- Several levels of accuracy
  - global or local
- Speedup based on:
  - Partitioning (hierarchy)
  - Event-Driven
  - Multi-rate
  - Simplified models
- Circuit Size and performance
  - up to 10M devices and 2 to 100x SPICE

FastSPICE - Algorithms

- Circuit Matrix Partitioning
- Event-driven evaluation
- Multi-rate
- RC network reduction
- Hierarchical-Isomorphic Partition
- Adaptive partitioning

FastSPICE - Component Modeling

- Transistors
  - Varied levels of detail available in FastSpice
  - Table models + simplified equations
    - I-V and Q-V tables
    - Device capacitance options:
      - linear/nonlinear, grounded only
- Coupling Capacitors
  - Ground all
  - Ignore below selected value
  - Constant value
  - Voltage-dependent for analog

Performance Gain

- Typical circuits used to evaluate performance:
  - Memory – best case:
    - Very high number of transistors
    - few blocks with very high level of repetitiveness
  - PLL – worst case:
    - Medium number of transistors
    - Many blocks with low repetitiveness

MPU Performance Gain (SRAM)

- SRAM 16k - 100k MOSFETs, 40k eqns.
  - 16k occurances of a 6T memory cell
  - Additional blocks: logic cells, sense amps
  - Xeon 3 GHz, 8 processors

Simulator	Time (Normlized)	Speedup
SPICE		
1 cpu	1	1
2 cpu	0.75	1.33
4 cpu	0.38	2.64
8 cpu	0.36	2.75
FASTSPICE	0.012	83

MPU Performance Gain (PLL)

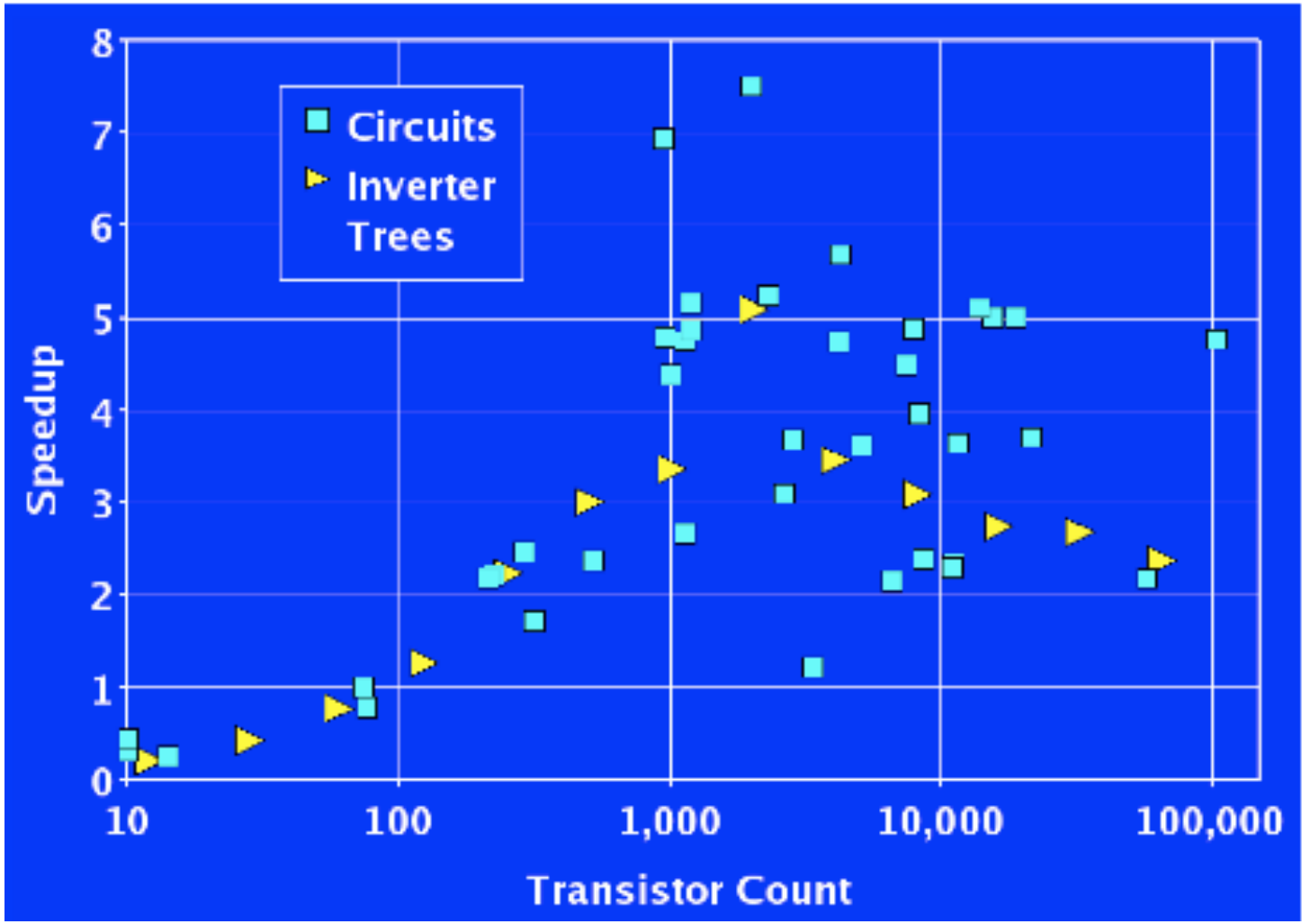
- PLL[1)] Layout extraction
  - 2200 MOSFETs, 14k Eqns, 21k Rs, 20k Cs
  - After RC reduction 10-30% less Rs and Cs

Simulator	Time (Normalized)	Speedup
SPICE		
1 cpu	1	1
2 cpu	0.55	1.8
4 cpu	0.27	3.8
8 cpu	0.24	4.2
FASTSPICE	0.24	4.2

After Partitioning/Parallelization number of iterations differs

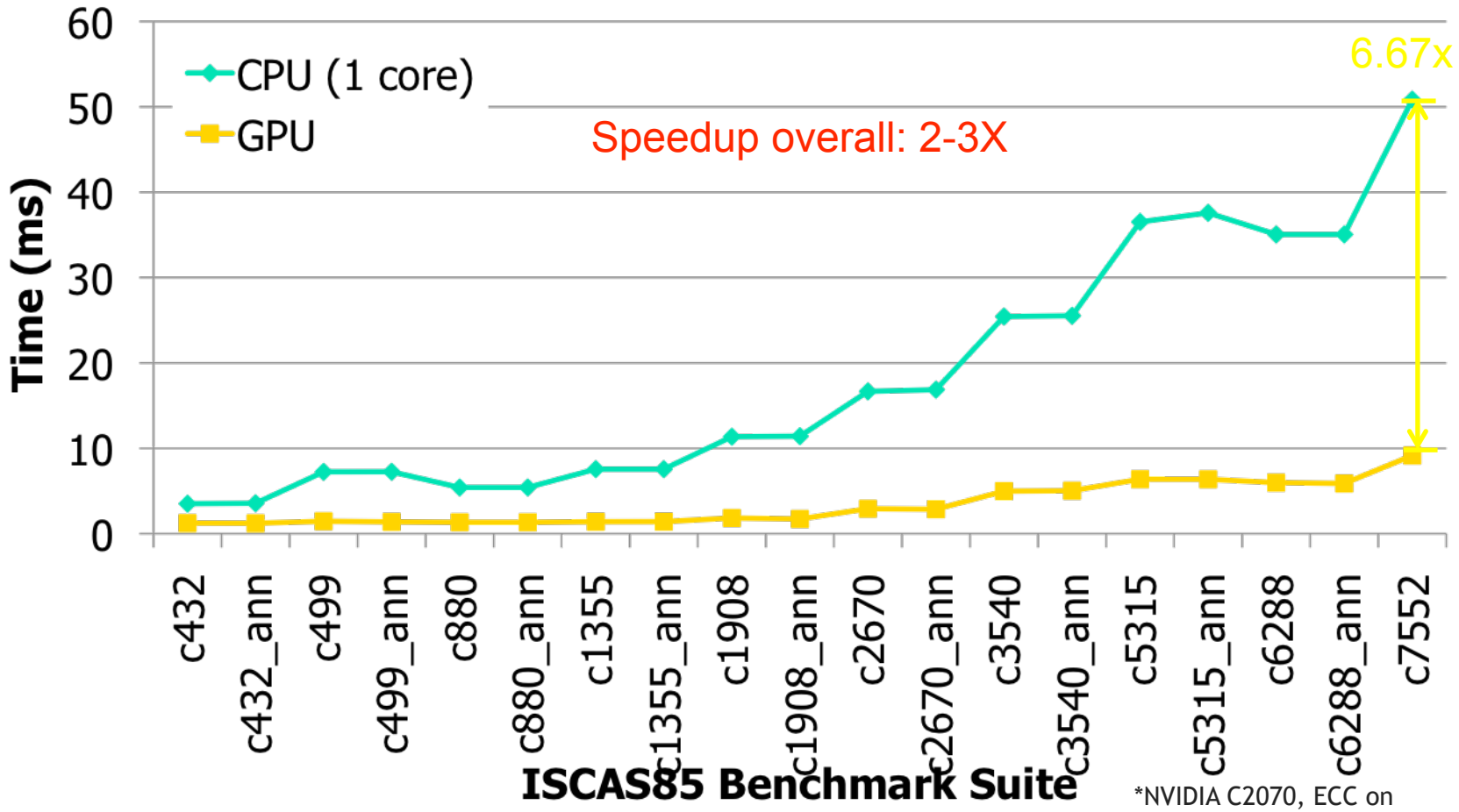
[1] Tabesh, M., Chen, J, Marcu, C, et al., A 65nm CMOS 4-Element Sub-34mW/Element 60GHz Phased-Array Transceiver, JSSC, vol 46, Dec. 2011

GPU Performance Gain – Model Evaluation only



R.E. Poore, GPU-Accelerated Time-Domain Circuit Simulation, IEEE-CICC, June 2009

GPU Performance Gain - BSIM4v7 Model Evaluation



*NVIDIA C2070, ECC on
*Intel X5690 (6 Core™) @ 3.47GHz

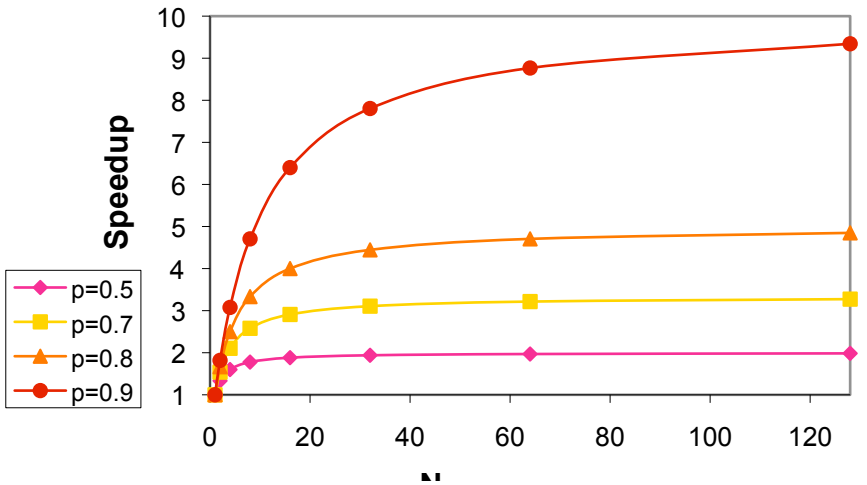
M.Naumov, F.Lannutti, et al., What does it take to accelerate SPICE on GPU, *GTC*, 2013

What limits the Speedup?

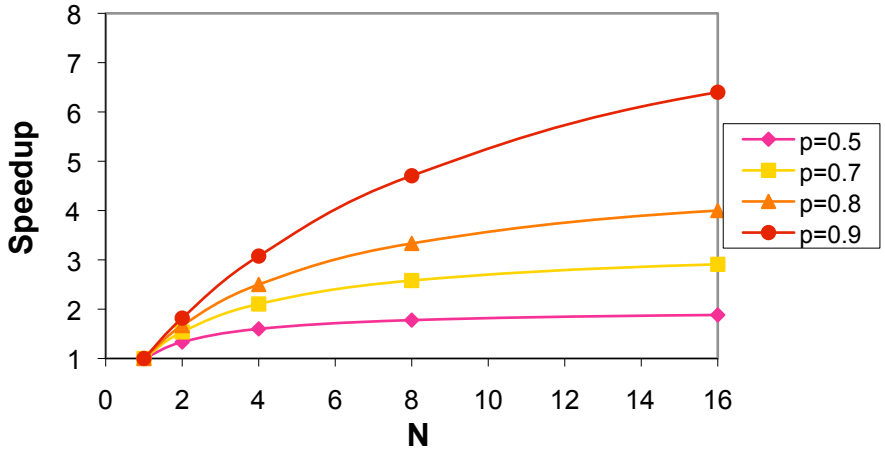
- Amdahl's Law

$$Speedup = \frac{T_s(=1)}{T_p} = \frac{s+p}{s+p/N} = \frac{1}{s+p/N}$$

s=0.1, p=0.9
s+p=1
Speedup=1/(0.1+0.9/N)



- Circuit Simulation
  - Optimal N = 4 - 8



Conclusion and Outlook

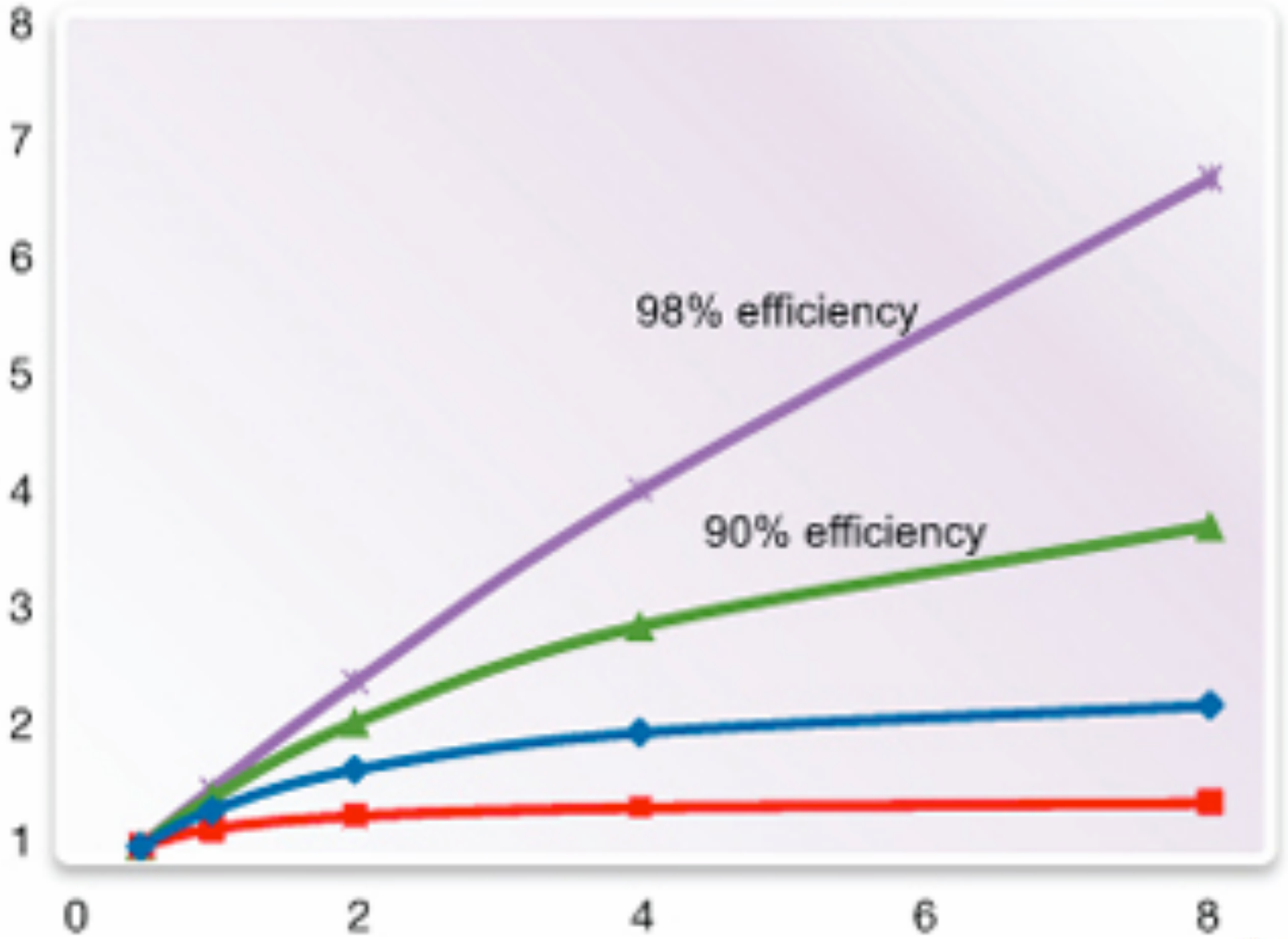
- Parallel HW offers a great opportunity for full-chip circuit (SPICE) simulation
  - Solution algorithms have been refined and adapted
  - Single simulation
    - Small number of processors (<16) offer best returns (Multi-core) due to saturation
    - Best result: 5X on 8 processors
  - Multiple simulations
    - linear performance scaling, Many-core or Cloud
- Speedup is circuit specific!
- More speedup
  - can only be obtained from novel intrinsically parallel algorithms

Conclusion and Outlook

- Future Progress will come also from Public-Domain Research
  - NGSpice – Linux open-source inspired effort
  - Xyce – Sandia Labs, public domain
- ESSCIR/ESSDERC 2014, MOS-AK Workshop
  - Open-Source CAD/EDA Tools – Parallel SPICE

Extra Slides

Performance Gain (EDA Marketing)



?

✔

FastSPICE - Hierarchy

- Derived from
  - User input
  - Automatically detected - Graph isomorphism
- Isomorphic hierarchical simulation
  - Repetitive structures - RAMs, standard cell designs
  - Time/memory savings simulating just one cell for all identical instances with identical inputs

FastSPICE - Hierarchy - Adaptive Partitioning

- Splitting - more representative cells are generated when voltages change
- Important speedup despite overhead