Circuit Simulation Update: GPU Progress; Electrothermal Cosimulation

September 2014

Rick Poore
Keysight EEsof EDA R&D

KEYSIGHT TECHNOLOGIES

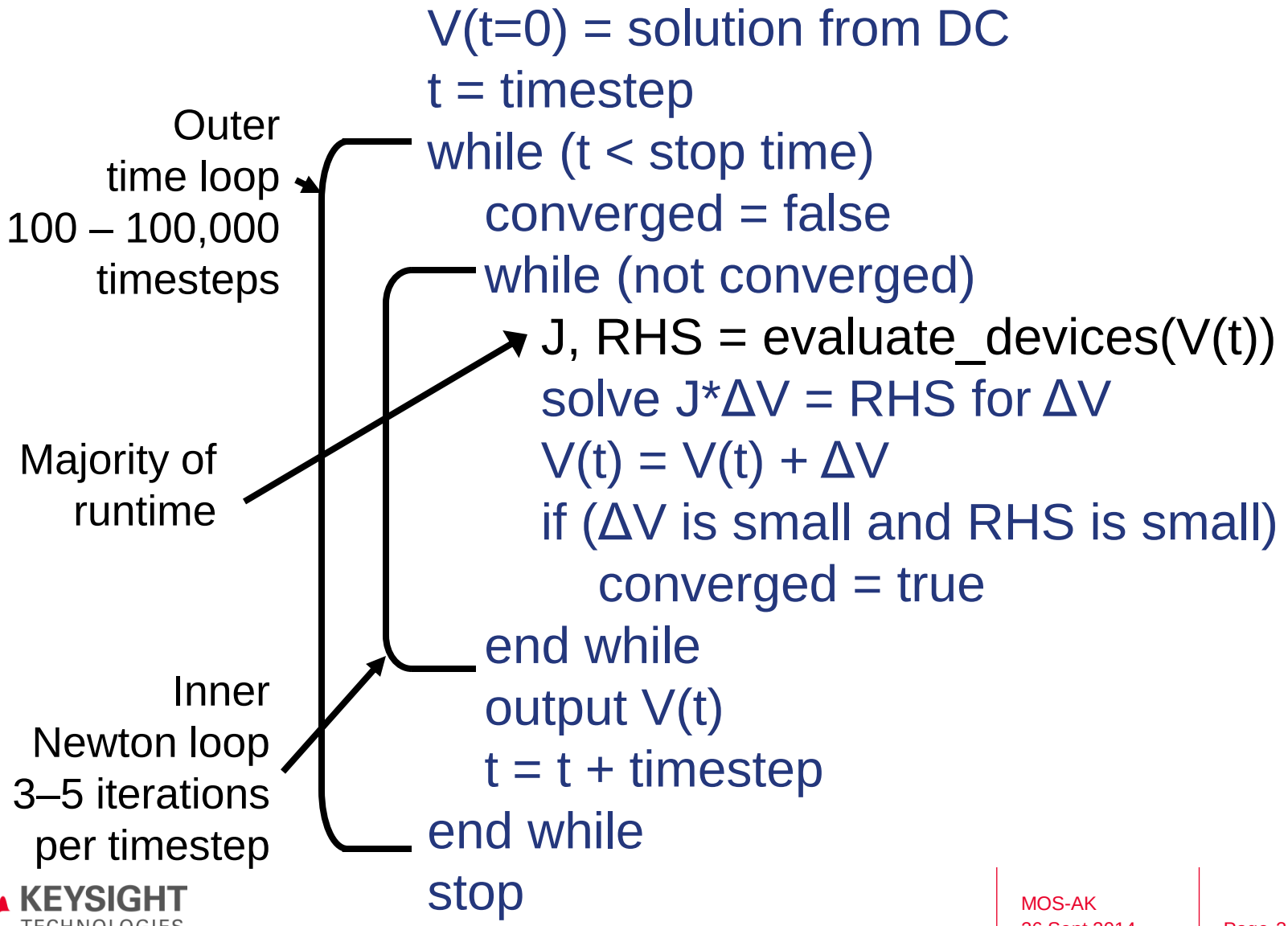
Outline

GPU time-domain circuit simulation: Where's the next bottleneck

- **Where does the time go**
- **GPU speedup**
- **Matrix factor and solve time**
- **GPU-based iterative solver**

Electrothermal cosimulation

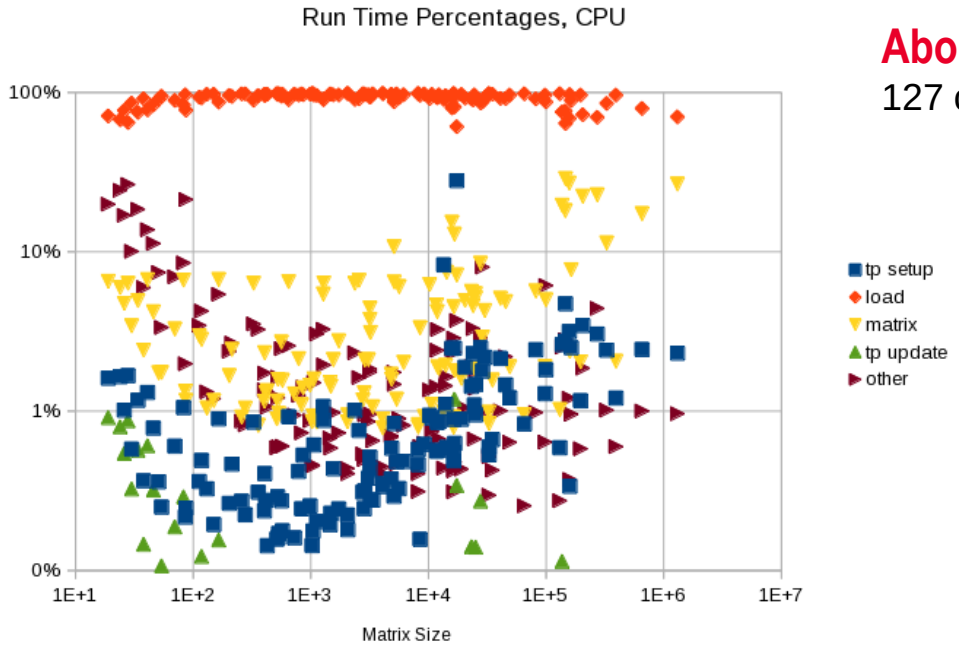
Transient Simulator Pseudocode



Where Does The Time Go

CPU time usage (main time loop, single threaded)

- 1% (0-28%) Pre-timepoint – a couple of circuits do a lot of convolution
- 91% (61-99%) Device evaluation (matrix load)
- 4% (1-29%) Matrix factor and solve (direct LU)
- 3% (0-27%) Everything else – output, timestep control, overhead



About the test suite

127 circuits of various sizes

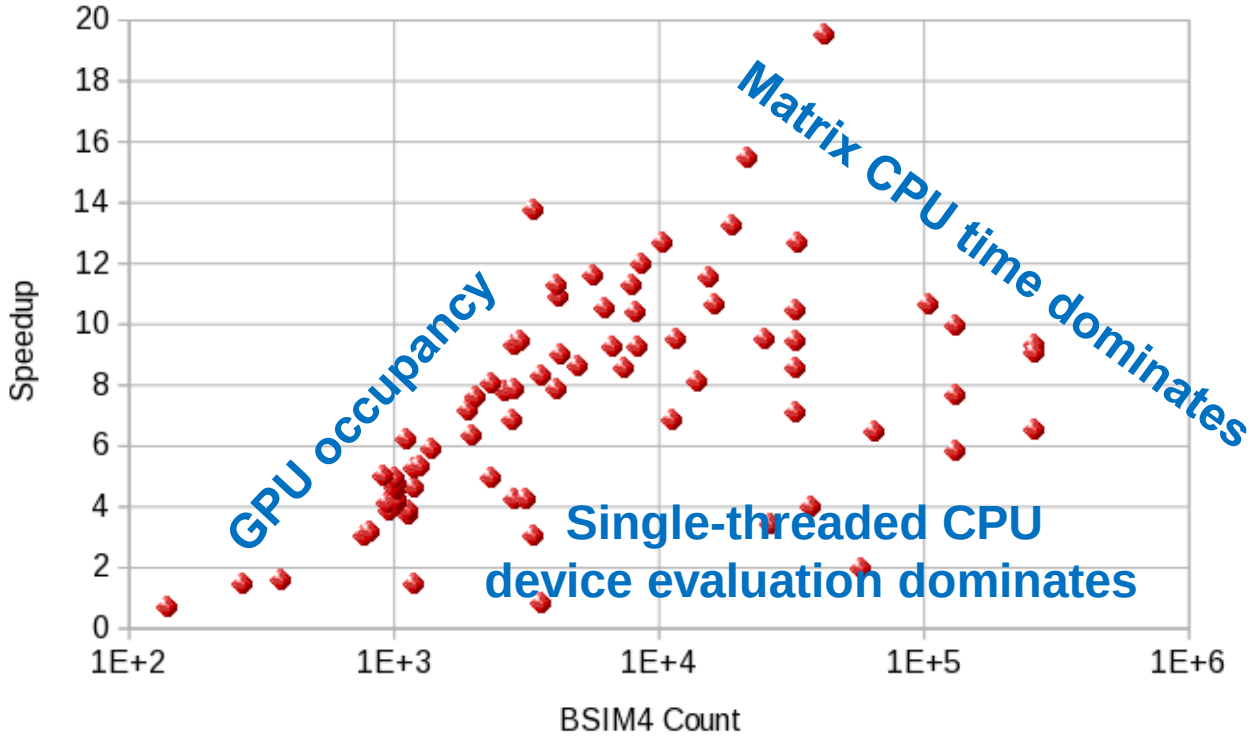
- 4 – 262k BSIM4s
- 24 – 2.75M total devices
- 19 – 1.31M matrix unknowns

GPU Speedup

Main time loop, versus CPU single threaded

– Device evaluation is 91% of CPU runtime on average

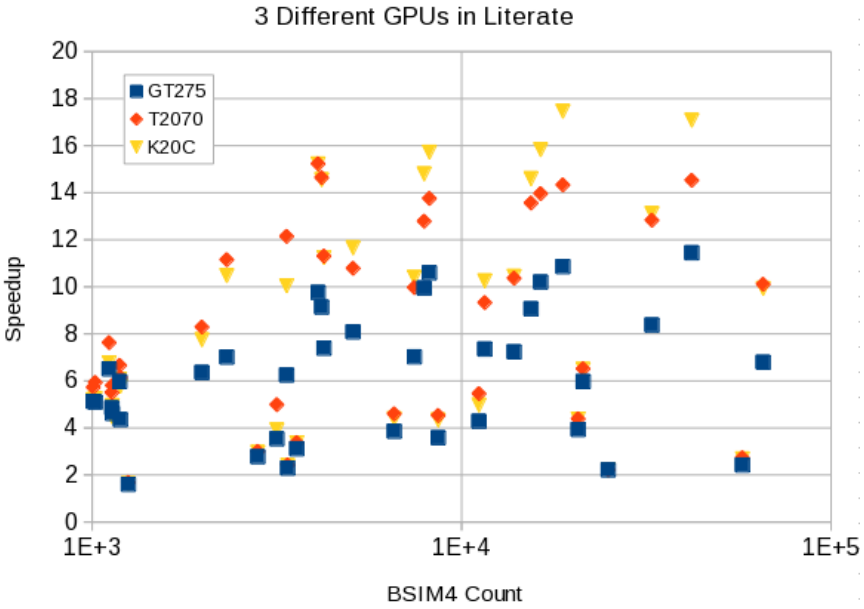
  • an obvious task to use a GPU to speed up



CPU: 2xE5-2667 v2 (32c, 3.3 GHz)
GPU: Nvidia K20c (2496c, 705 MHz)

GPU Speedup

Comparison of different GPUs with same CPU

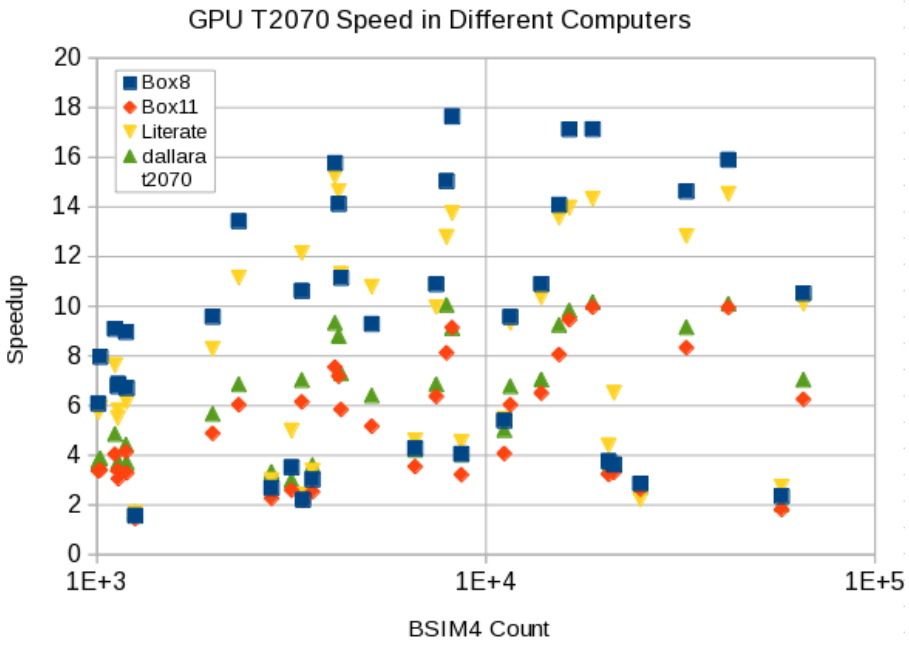


GPU	GPU Speedup	GPU Cores x Clock	Gflop/s (sp)	Gflop/s (dp)	Memory Bandwidth (GB/s)
GT275	6.1 ± 2.7	240 x 1458 MHz	700	88	127
T2070	8.2 ± 4.2	448 x 1147 MHz	1028	514	144
K20c	8.3 ± 4.8	2496 x 705 MHz	3520	1173	208
CPU: E5-1620 (4c, 3.6 GHz)					

- BSIM4 for GPU is coded mostly in single precision
- Single precision won't work well for other models (e.g. PSP)

GPU Speedup

Comparison of same GPU with different CPUs

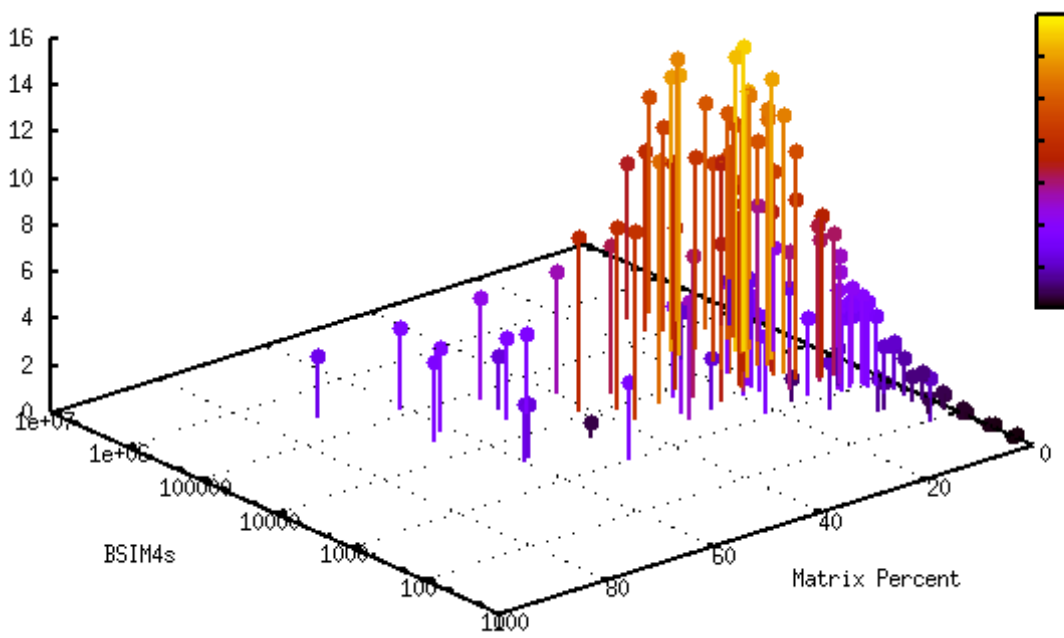


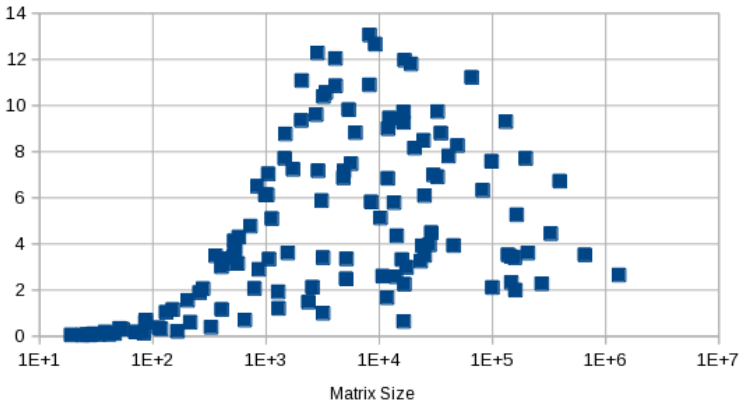
Computer	GPU Speed	CPU Speed	CPU
Box8	8.8 ± 4.8	1.00	Intel Q6600
Literate	8.2 ± 4.2	1.59	E5-1620
Box11	5.0 ± 2.4	2.31	AMD FX8320
Dallara	5.7 ± 2.6	2.33	Intel E5-2667
GPU: Nvidia T2070			

GPU Speedup

Main time loop, versus CPU single threaded

– As the matrix size goes up, GPU speedup goes down as we start to be limited by the single-threaded CPU matrix solve





Where Does The Time Go

GPU time usage (main time loop, single threaded)

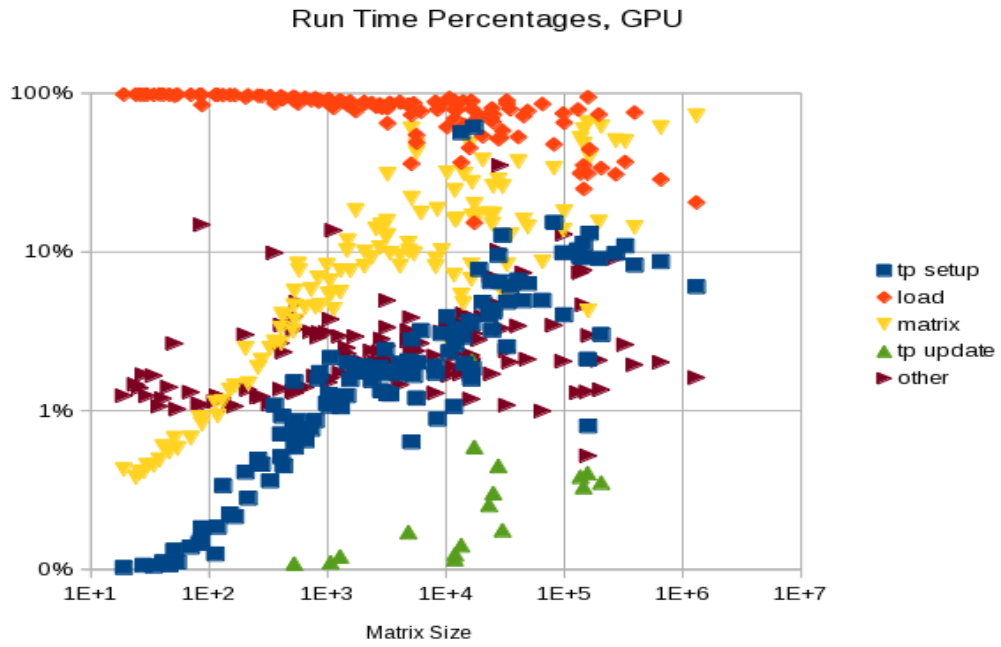
1% – 4% (0-61%) Pre-timepoint – a couple of circuits do a lot of convolution

91% – 78% (15-98%) Device evaluation (matrix load)

4% – 15% (0-72%) Matrix factor and solve

3% – 3% (1-35%) Everything else – output, timestep control, overhead

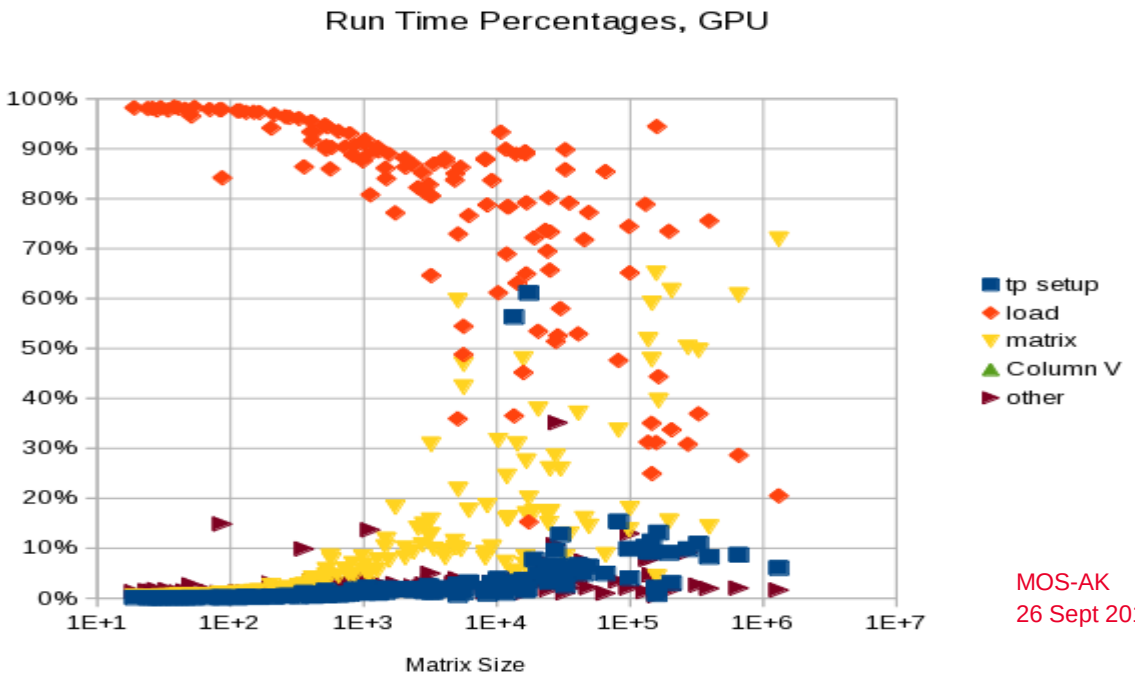
CPU



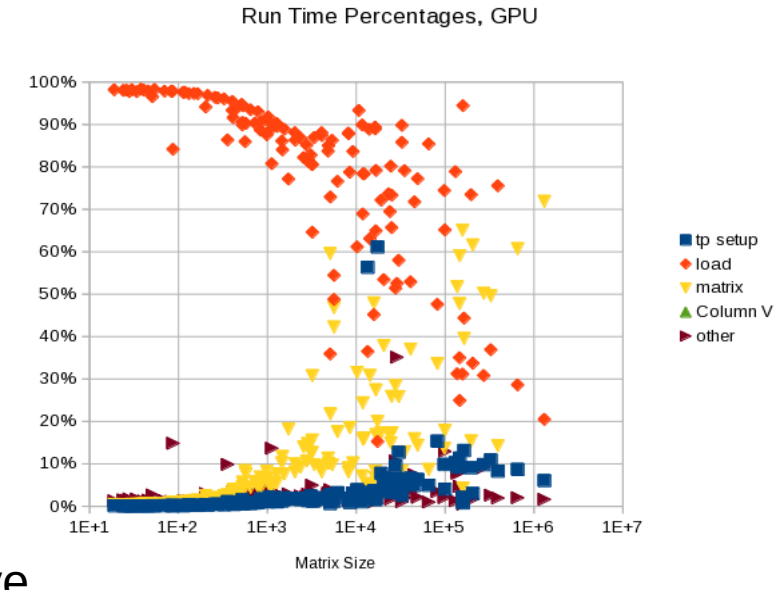
Matrix Time Starts To Dominate

GPU time usage (main time loop, single threaded)

- 15% (0-72%) Matrix factor and solve
- Direct sparse matrix packages, especially those optimized for circuit simulation (e.g. KLU), are single threaded
- Generic threaded sparse matrix packages tend to be slower than a good circuit-simulation specific single-threaded solver



GPU Matrix Solver



- No public, direct sparse matrix solvers targeted for GPU
- Iterative solvers, anyone?
- GMRES used in harmonic balance to solve systems with tens of millions of unknowns
- GPU based iterative solvers (open source):
  - ViennaCL (http://viennacl.sourceforge.net/)
  - CUSP (http://cusplibrary.github.io/)
  - Paralution (http://www.paralution.com/)

Commercial iterative GPU solvers

- Nvidia's AmgX (https://developer.nvidia.com/amgx)
- MatrixPro-GSS (http://www.matrixprosoftware.com/products/products.shtml)
- CULA sparse (http://www.culatools.com/sparse/)

GPU Iterative Matrix Solver

- Iterative solvers generally require a good preconditioner
- Preconditioner is some approximation of the inverse of the matrix
- In harmonic balance, using an inverse based on just the DC component (diagonal preconditioner) is efficient
  - DC matrix is small compared to full matrix of 5-50 complex harmonics
- Transient matrix has no easy to compute subset
- Time for sending the matrix and RHS to the GPU, solving, and returning the correction vector to the CPU must be faster than just running a single-threaded solver on the CPU

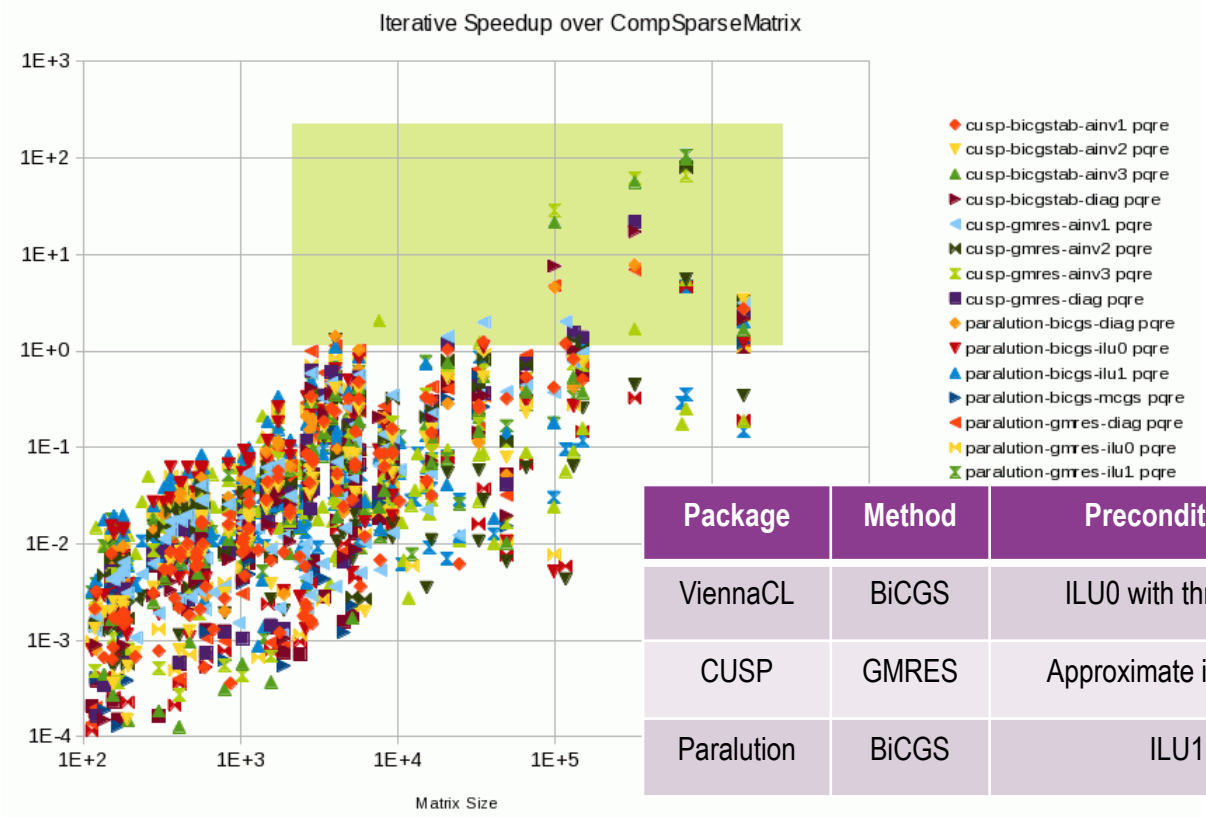
GPU Iterative Matrix Solver

Test methodology

- Solve the system:
  x = 1
  b = Ax
  solve Ax=b for x
  successful if $\frac{\|x-1\|_2}{\sqrt{N}} < 0.01$ (at least 2 digits of accuracy)
- Preconditioner construction time not included in solve time
  - Assumes we'll be able to find a preconditioner that can be used for multiple iterations and timesteps
- Test suite of 451 matrices from transient circuit simulation
  - Best combination: 161 matrices solved successfully
- Computer: CPU=E5-1620, GPU=Nvidia K20c

GPU Iterative Matrix Solver

– Speedup of various GPU iterative solvers and preconditioners versus our CPU single-threaded direct solver

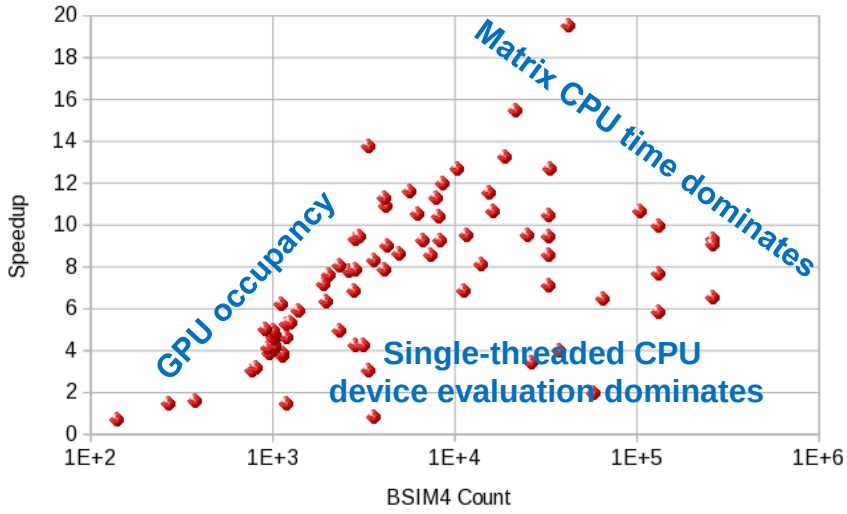


Package	Method	Preconditioner	Count	Speed > 0.5
ViennaCL	BiCGS	ILU0 with threshold	164	5
CUSP	GMRES	Approximate inverse 1	120	9
Paralution	BiCGS	ILU1	111	12

Conclusion

What's next

- Newer and faster GPUs
- Continue to move more of the transient algorithm to GPU

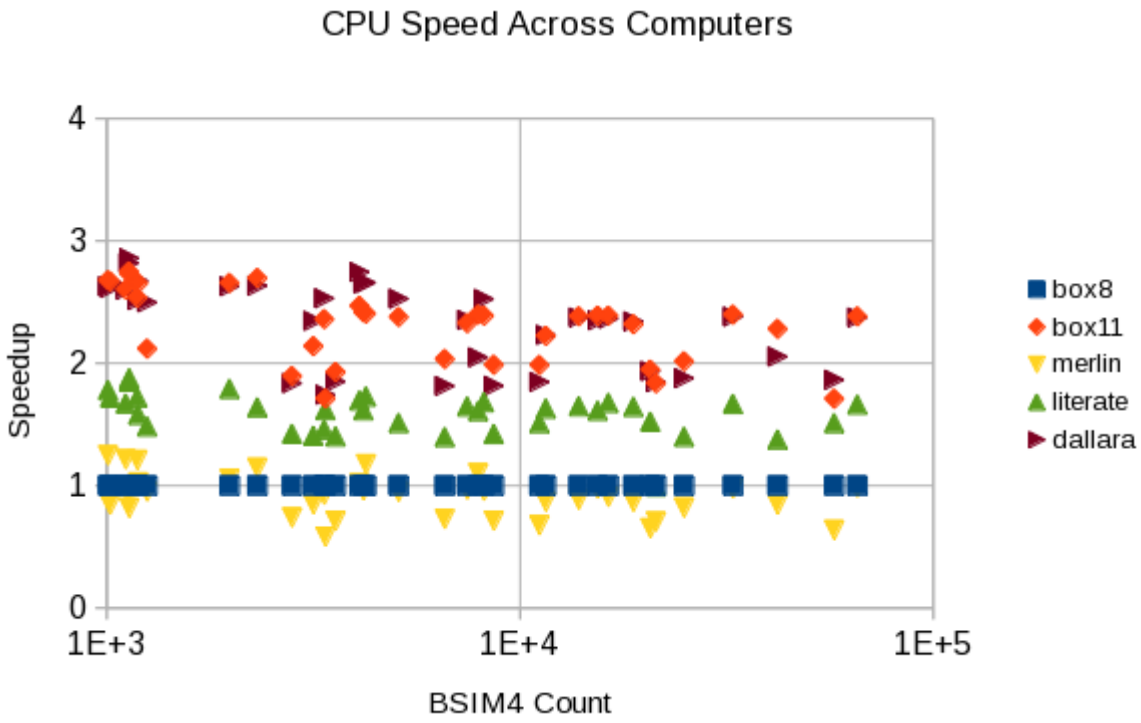


- Address the matrix factor and solve bottleneck
  - GPU-based iterative solver (e.g. BiCGS, GMRES)
  - Is there a good preconditioner than can be used for multiple iterations and timesteps?

CPU Speedup

Careful, CPUs are getting faster as well!

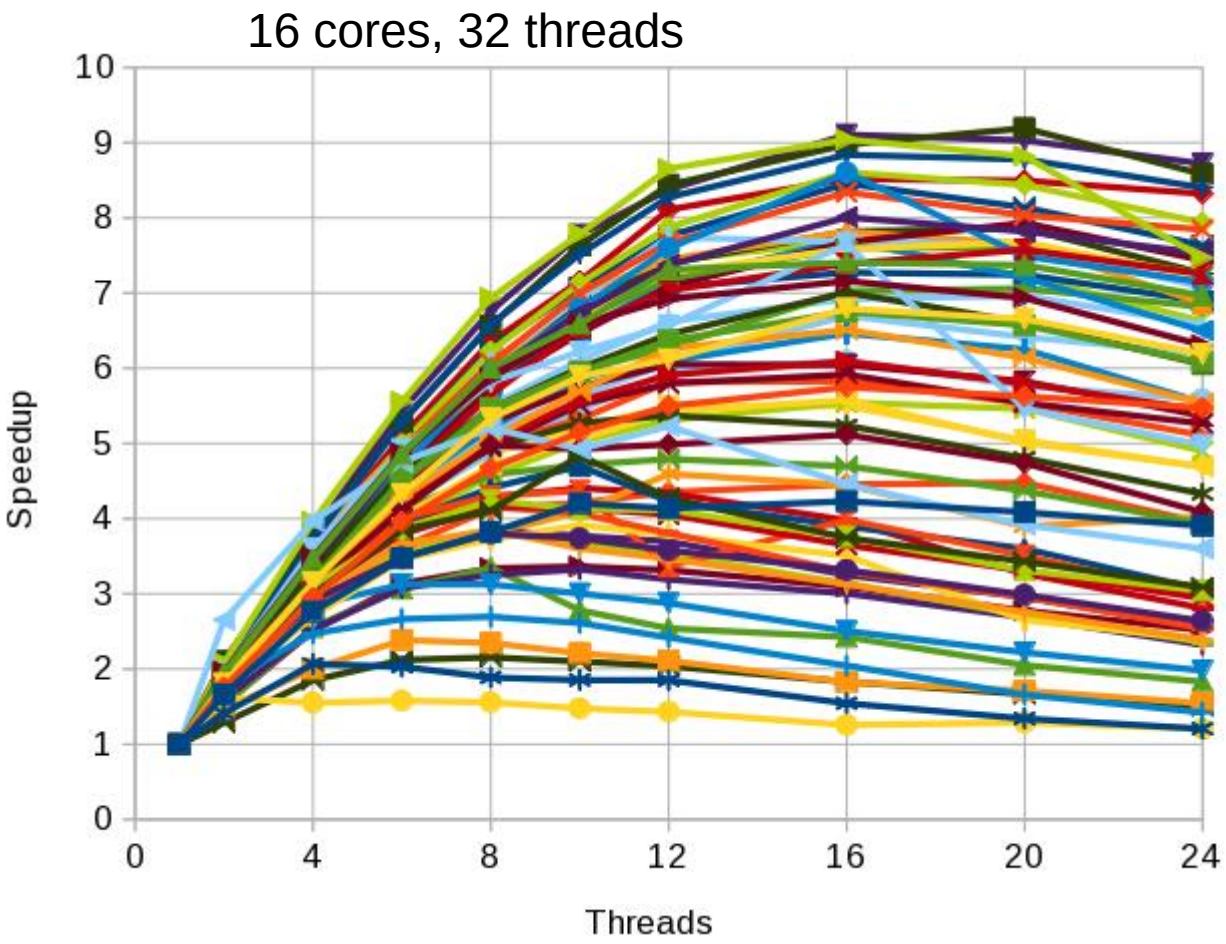
– Comparison of different single threaded CPUs over the years



Computer	Speed	CPU
Merlin	0.92 ± 0.17	AMD Opt 6174
Box8	1.00 ± 0.00	Intel Q6600
Literate	1.59 ± 0.17	Intel E5-1620
Box11	2.31 ± 0.29	AMD FX8320
Dallara	2.33 ± 0.34	Intel E5-2667

CPU Threading Speedup

Subset of 68 circuits (mostly the larger)



Threads	Speedup	Serial Code
2	1.8 ± 0.2	9.8%
4	3.1 ± 0.4	9.7%
8	4.9 ± 1.2	9.2%
12	5.5 ± 1.9	10.6%
16	5.7 ± 2.2	12.1%
20	5.4 ± 2.3	14.2%
24	5.0 ± 2.2	16.5%

Amdahl's Law

$$R = \frac{1}{S + (1 - S)/N}$$

R = speedup ratio
S = fraction of serial code
N = number of threads

CPU: 2xE5-2667 v2 (32c, 3.3 GHz)
GPU: Nvidia K20c (2496c, 705 MHz)

Outline

GPU time-domain circuit simulation: Where's the next bottleneck

Electrothermal cosimulation

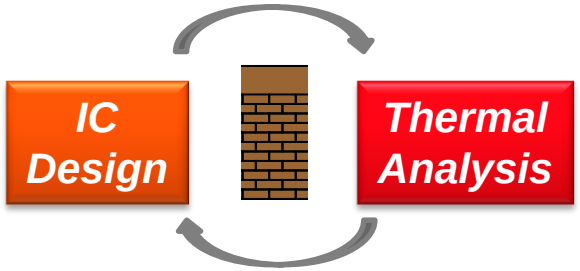
- **Integrating electrical and thermal simulator**
- **Example: MMIC PA**

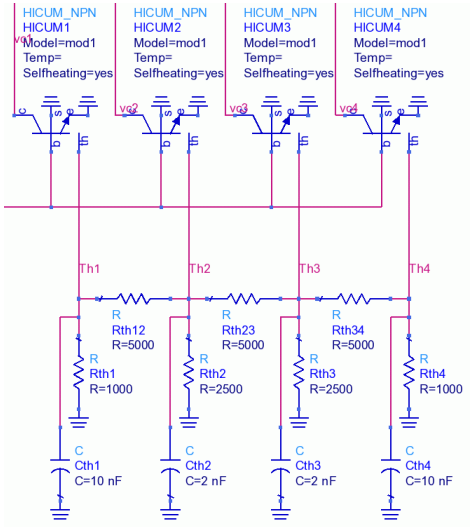
KEYSIGHT TECHNOLOGIES

The Problem

On-chip devices aren't at ambient temperature

- High power devices + high levels in integration
  = on-chip temperature rise
- IC performance depends on device temperature
- How do we determine device temperature?
  - ~~Assume all devices at $T_{AMBIENT}$~~
  - Stand-alone thermal simulator
  - Device models with self-heating
  - Attach thermal coupling network

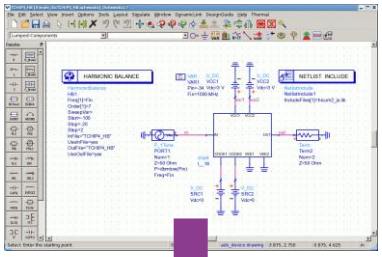


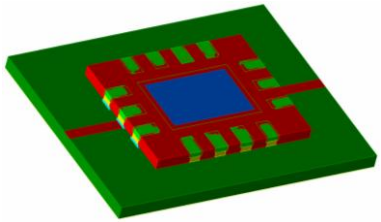


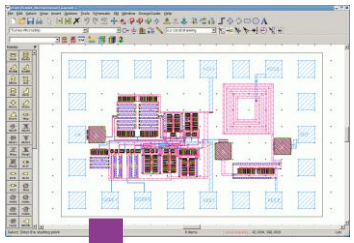
Limitations of Existing Solutions

- Stand-alone thermal simulator
  - Manual interchange of powers and temperatures
- Device models with self-heating model
  - Not all models have self heating
  - Requires accurate extraction of R_{TH} and C_{TH}
  - Does not include thermal coupling between devices
  - Does not include impact of packaging
- Attach thermal coupling network
  - Requires devices with thermal nodes
  - Large device count makes slow extraction of RC network
  - Does not account for nonlinear thermal properties k(T)

New Approach – Full Electro-Thermal Simulation







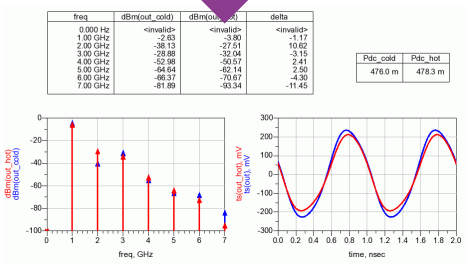
Thermal technology files

Circuit Simulator
Read temperatures
Solve electrical equations
Write power dissipation

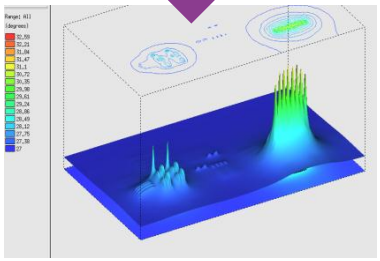
$T_{DEVICES}$

P_{DISS}

Thermal Simulator
Read power dissipation
Solve thermal equation
Write temperatures



Iteration loop is done automatically until powers and temperatures are self-consistent

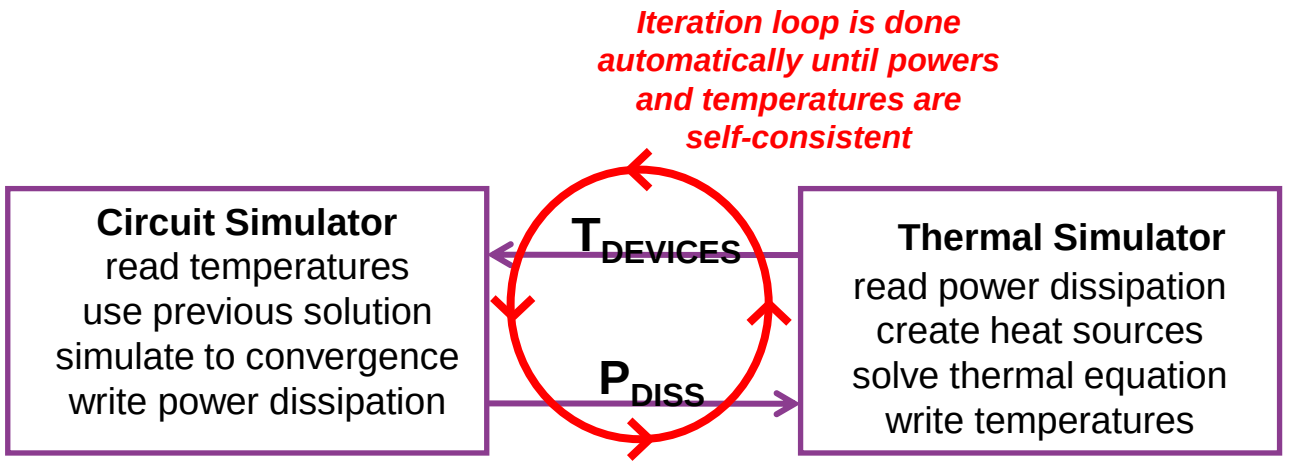


3-D FEM Cosimulation: DC, AC/SP, HB

– Assumes no time-dependence (DC, AC/SP)

– Assumes frequencies faster than thermal time constants (HB)

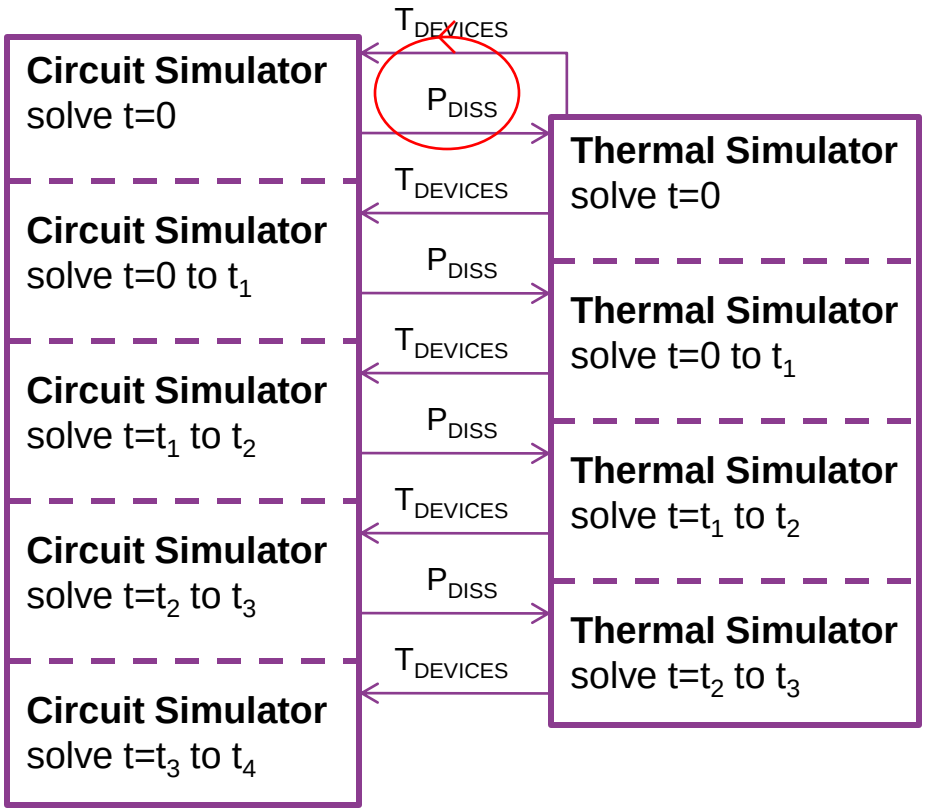
- Valid for high frequency 1-tone HB (> ~10-100 MHz)
- Valid for multitone HB where all mixing products are high frequency (> ~10-100 MHz)
- Not valid for 2-tone IMD testing (using Envelope HB)



3-D FEM Cosimulation: Transient, Envelope

- Loosely-coupled cosimulation between electrical and thermal simulator
- Circuit simulator computes average power dissipation for devices over a small timestep
- Thermal simulator computes average device temperatures over the same timestep
- Various strategies to control size of timestep – small enough step for consistency

each simulator uses its own choice of timestep within its simulation

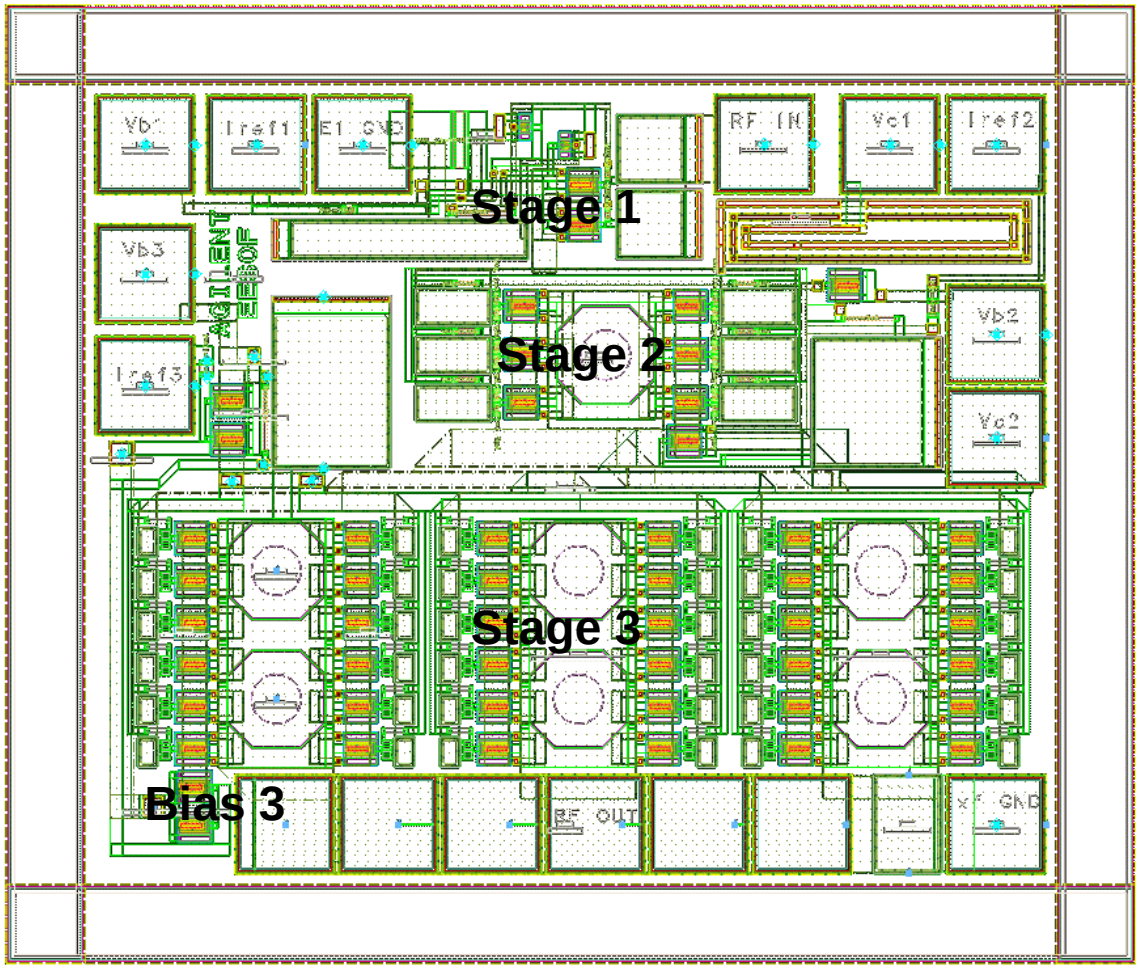


Electrothermal Implementation Details

- Electrothermal iteration loop goes around existing simulator
- Iteration done only on fully converged electrical solution
- Modify circuit simulator:
  - All devices compute and output P_{DISS}
  - Read new device temperatures and recompute temperature scaling
  - Average P_{DISS} over a thermal timestep for transient
- Need mapping from circuit simulator device name and P_{DISS} to layout location of heat source
- Only significant convergence problem – bipolar thermal runaway
  - Limit temperature change per iteration (clip or damp)

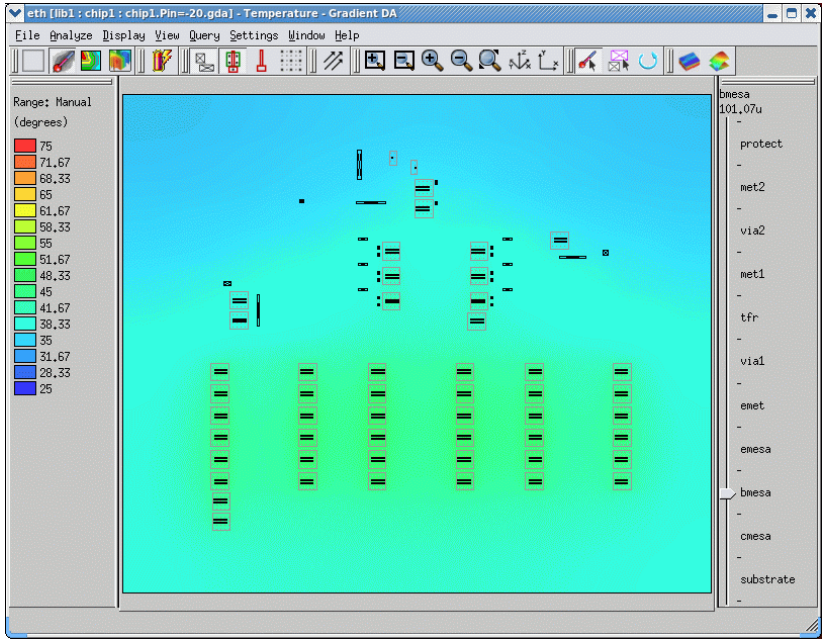
Example: HB Simulation of MMIC PA

- 3-stage PA in III-V HBT WinSemi H02U-01
- $P_{DISS,DC}$ = 0.44W
- $P_{DISS,RF}$ = 0.44 – 2.10 W
- $P_{OUT,RF}$ up to +30.7 dBm

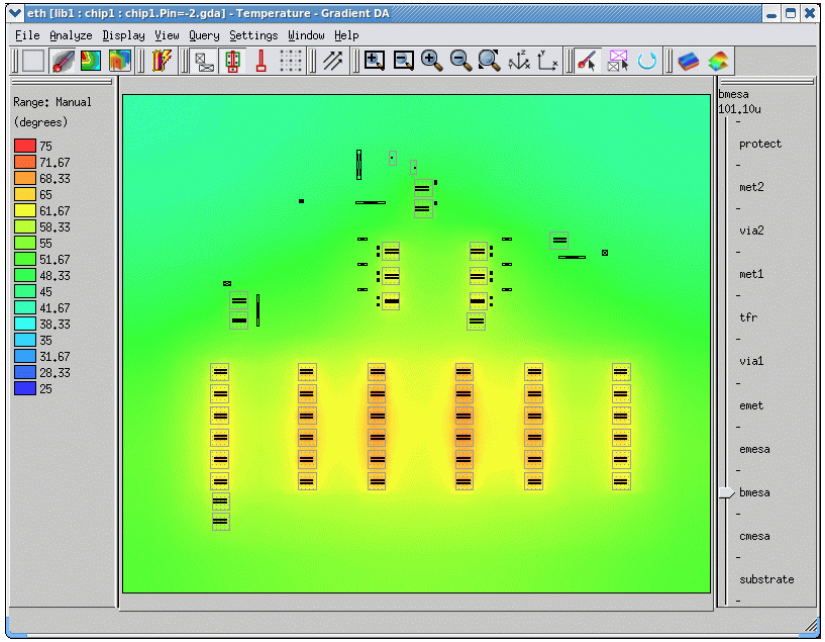


Resulting Temperatures at Various Power Levels

P_{IN} = –20 dBm
P_{DISS} = 0.46W

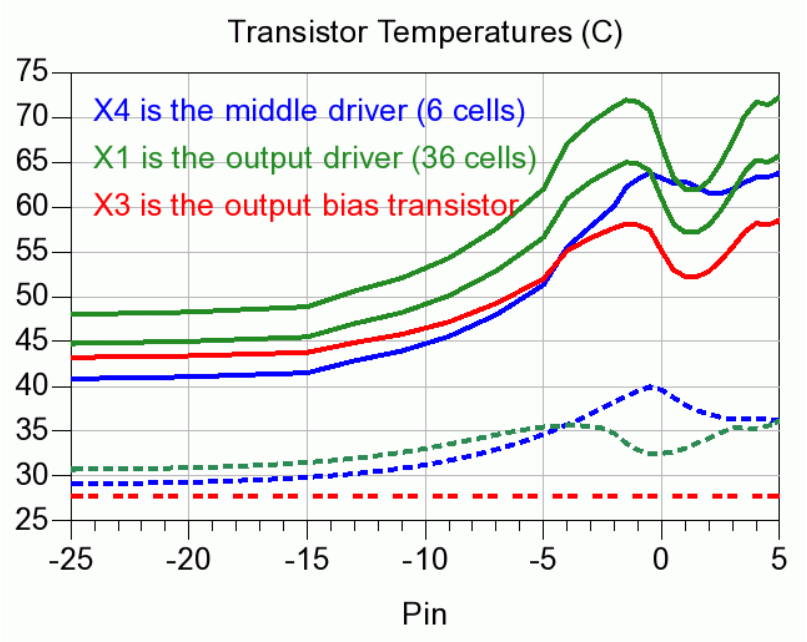


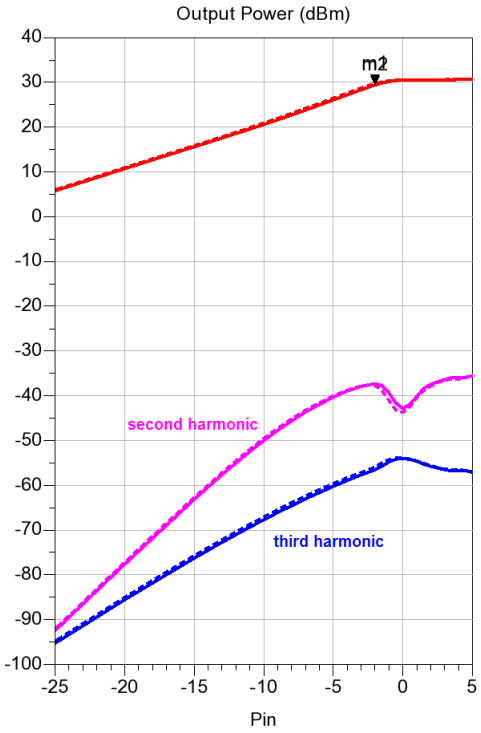
P_{IN} = –2 dBm
P_{DISS} = 1.73W



Electrical and Thermal Results

- Top traces electrothermal $R_{TH,BOT}$=30 K/W
- Temperatures of 2 of 36 different output stage cells
- Bottom traces self heating
- 25 point HB sweep took 6 minutes to run





Questions?

– GPU simulation

– Electrothermal simulation