OpenCL Sparse Linear Solver for Circuit Simulation

Jason Mak, Matthew Guthaus

University of California, Santa Cruz

Motivation

• Sparse linear systems are common in scientific and engineering problems.

• Computing solutions for large systems may be extremely time consuming.

• Consumer graphics hardware , with its innate parallel architecture and numerous processing cores, has shown success in speeding up such computations.

• Nodal analysis of circuits, which may contain millions of nodes , generates very large, sparse systems.

•Multiple libraries have been written to use a GPU as a parallel linear solver. This project tests the effectiveness of one such library in circuit analysis.

$$A\,x = b$$

The Conjugate Gradient Algorithm

For j=1 to (max iterations), do

(a) $\alpha_j := (r_j, r_j) / (Ap_j, p_j)$

(b) $x_{j+1} := x_j + \alpha_j p_j$

(c) $r_{j+1} := r_j - \alpha_j A p_j$

(d) $\beta_j := (r_{j+1}, r_{j+1}) / (r_j, r_j)$

(e) $p_{j+1} := r_{j+1} + \beta_j p_j$

• CG is an iterative method that takes multiple, repeated steps to allow the solution vector x to converge to the correct value.

• Most of computation is spent on two operations: matrix-vector multiplications (a) and vector inner products (a and d).

• Both operations can naturally be parallelized.

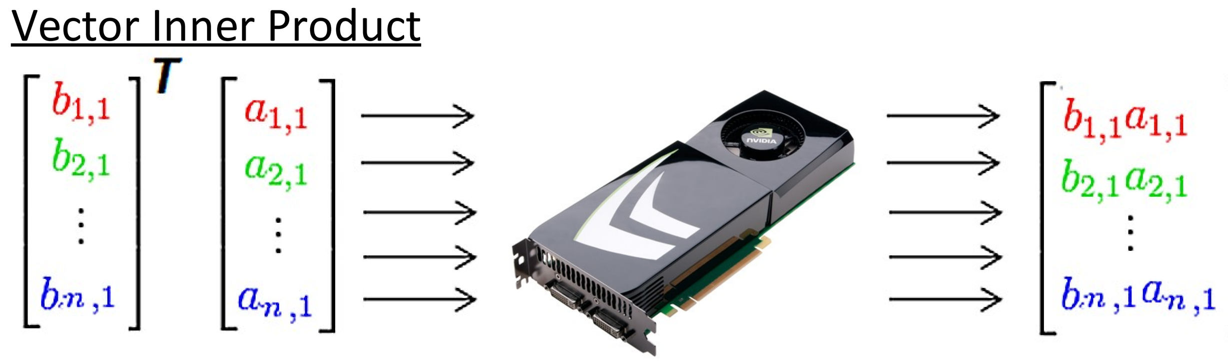
• In transient circuit analysis, the entire algorithm may be run repeatedly to simulate a circuit's behavior over time.

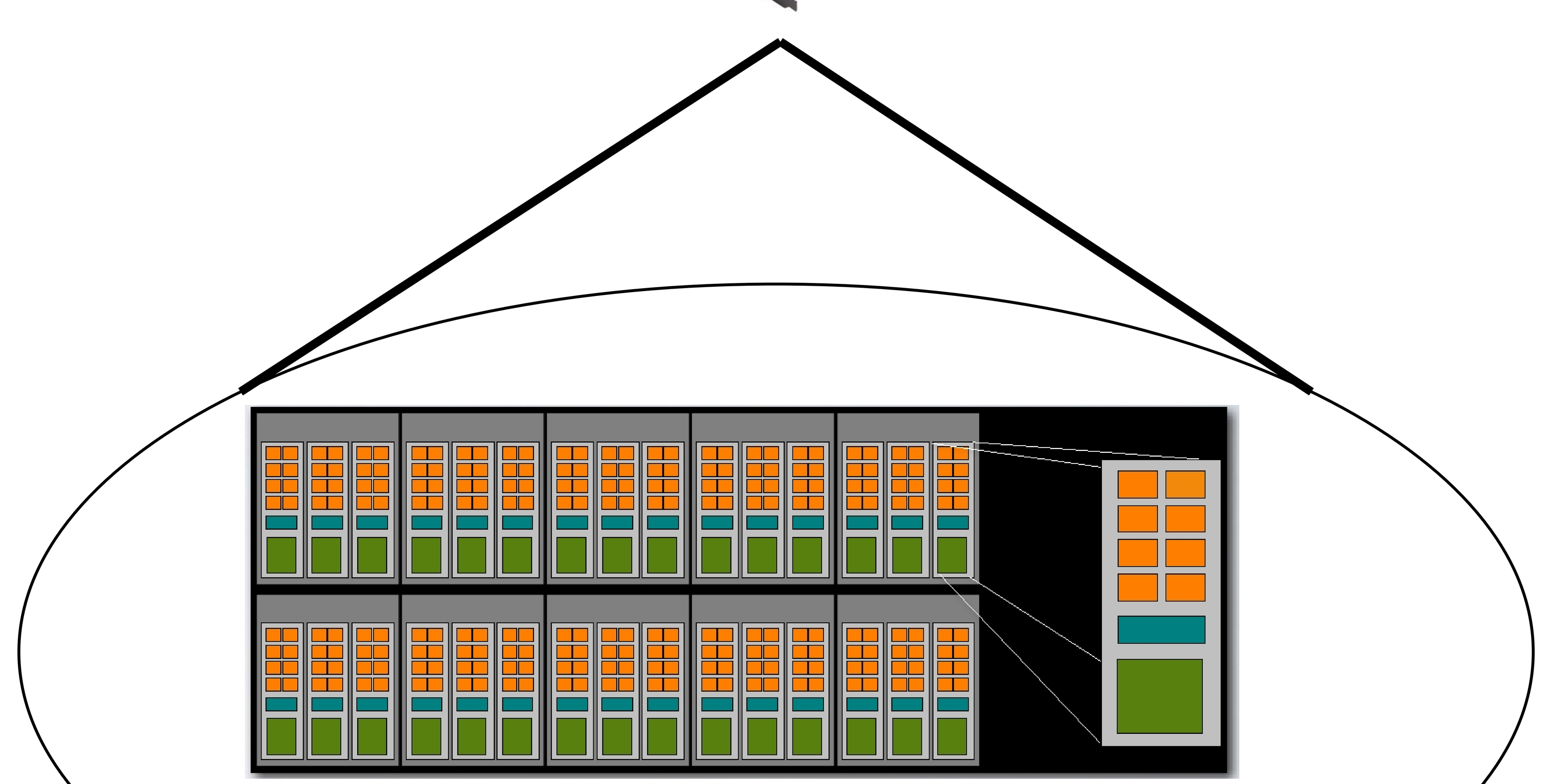
Matrix Vector Multiplication

Ex

$$\begin{bmatrix} 1 & 0 & 2 & 0 \\ 0 & 3 & 0 & 4 \\ 0 & 0 & 5 & 0 \\ 6 & 0 & 0 & 7 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 5 \\ 1 \\ 8 \end{bmatrix} = \begin{bmatrix} 4 \\ 47 \\ 5 \\ 68 \end{bmatrix}$$

Vector Inner Product





GPUs are known to possess many processing cores (shown in orange). The GTX 285 has 240 cores.

OpenCL and ViennaCL

•OpenCL is both an API and programming language (based on C) designed for making parallel applications.

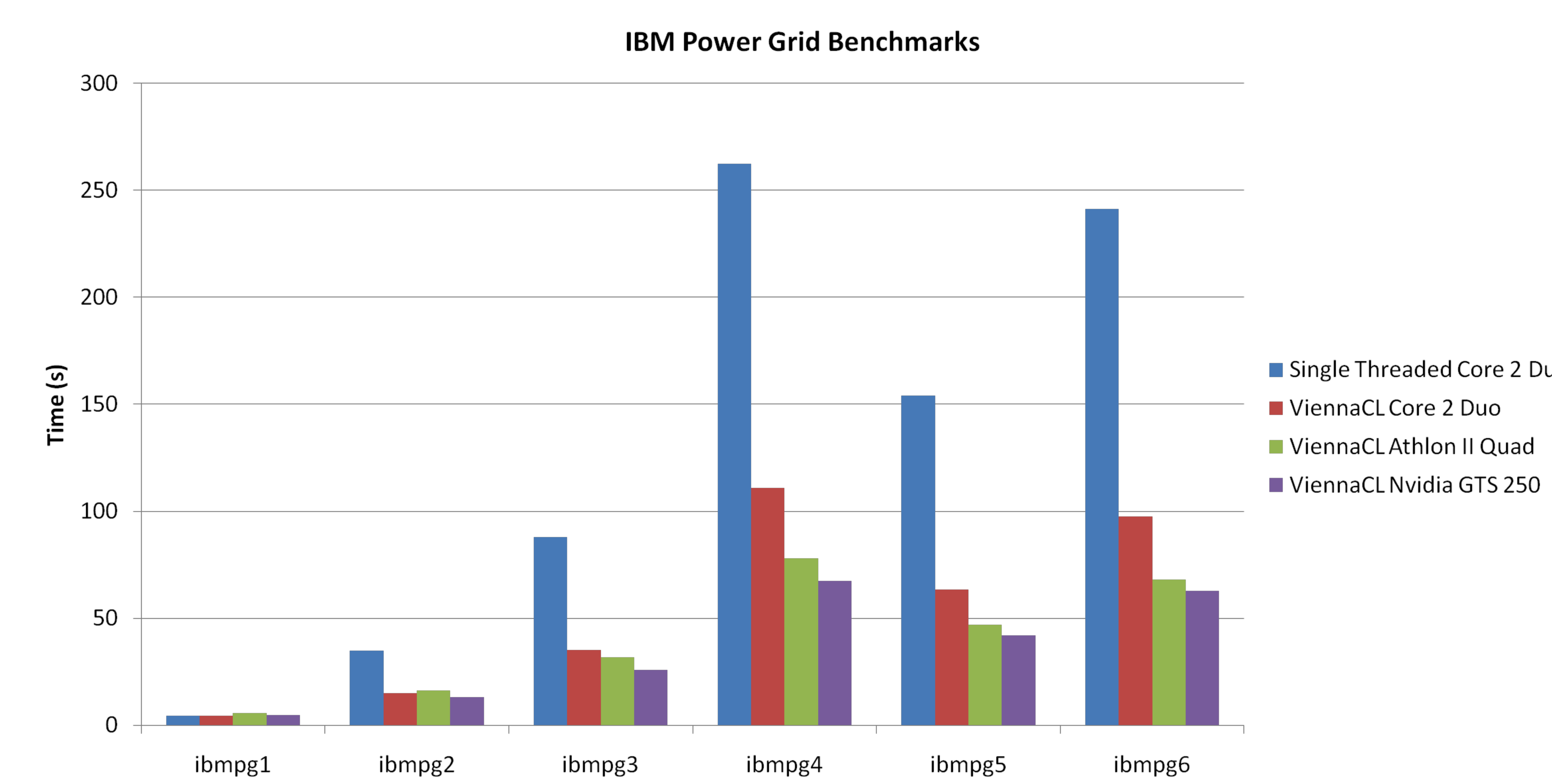
• OpenCL supports both CPUs and GPUs.

• ViennaCL is an open source library created by the Vienna University of Technology that implements linear solvers using OpenCL.

• ViennaCL employs black-box solvers by wrapping its OpenCL implementations in high-level interfaces to hide the tedious intricacies of GPU parallel programming from the user.

Power Grid Circuit Benchmarks

Name	current (I)	nodes	resistors	shorts (zeroes)	voltage sources	metal levels
ibmpg1	10774	30638	30027	14208	14308	2
ibmpg2	37926	127238	208325	1298	330	5
ibmpg3	201054	851584	1401572	461	955	5
ibmpg4	276976	953583	1560645	11682	962	6
ibmpg5	540800	1079310	1076848	606587	539087	3
ibmpg6	761484	1670494	1649002	836107	836239	3



Conclusion

• Both a single threaded implementation and ViennaCL's implementation of the conjugate gradient method were tested on IBM power grid circuits.

•ViennaCL's implementation yielded a speedup on both CPUs (Intel Core 2 Duo and Athlon II) and a GPU (Nvidia GTS 250).

• In some cases, the GPU yielded a speedup by a factor of 4.

• The black-box linear solver was successful in speeding up circuit analysis and could be an easy replacement for linear solvers in in other scientific applications as well.

UC SANTA CRUZ

VLSI Design Automation