CUSPICE

The revolutionary NGSPICE on CUDA Platforms

Francesco Lannutti\textsuperscript{1,2}, Francesco Menichelli\textsuperscript{1}, Mauro Olivieri\textsuperscript{1}
\textsuperscript{1} ”Sapienza” University of Rome, DIET
\textsuperscript{2} NGSPICE Team

Sep 26\textsuperscript{th}, 2014 – MOS-AK 2014, Venice – Italy
Agenda

Time Consuming Parts of SPICE

The Stamp Method

CUSPICE Features

Device Model Evaluation in the GPU

Circuit Matrix and RHS Update

Topology Matrix Method

Local Truncation Error in the GPU

Simulation Speedup

Conclusion
Time Consuming Parts of SPICE

- **OP Analysis**
  - Device Model Evaluation
  - Linear System Solution

- **Transient Analysis**
  - Device Model Evaluation
  - Linear System Solution
  - Local Truncation Error and Time Step Correction

- **Device Model Evaluation**
  - Takes about 40%-70% of the entire simulation time
  - It depends on the circuit size and complexity
From Circuit to Netlist to Matrix

Circuit:

Netlist (text file):

<table>
<thead>
<tr>
<th></th>
<th>V_k</th>
<th>V_j</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>R2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>R3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>R4</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>V1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Circuit Matrix:

\[
\begin{bmatrix}
G_1 & -G_1 & 0 & 1 \\
-G_1 & G_1 + G_2 + G_3 & -G_3 & 0 \\
0 & -G_3 & G_3 + G_4 & 0 \\
1 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
V_1 \\
V_2 \\
V_3 \\
I_{xs}
\end{bmatrix}
= \begin{bmatrix}
0 \\
0 \\
0 \\
V_s
\end{bmatrix}
\]

KCL + Device Models Laws:

\[ G \times V = I_s \]

\[ \sum V = V_s \]

Resistor Stamp

Sparse Matrix | RHS
---|---
V_k | V_j
k | G | -G
j | -G | G

Voltage Source Stamp

<table>
<thead>
<tr>
<th></th>
<th>Sparse Matrix</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>V_k</td>
<td>V_j</td>
<td>I_{xs}</td>
</tr>
<tr>
<td>k</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>j</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>xs</td>
<td>1</td>
<td>-1</td>
</tr>
</tbody>
</table>
The Matrix Updates: Stamp Method

In order to guarantee O(1) Circuit Matrix and RHS update

Every device model stores pointers to the exact locations in the Circuit Matrix and RHS where it has to write data

Matrix update for device model (the first instance of resistor R₁)

\[
\begin{bmatrix}
G₁ & -G₁ & 0 & 1 \\
-G₁ & G₁ + G₂ + G₃ & -G₃ & 0 \\
0 & -G₃ & G₃ + G₄ & 0 \\
1 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
V₁ \\
V₂ \\
V₃ \\
Iₓₛ
\end{bmatrix}
= \begin{bmatrix}
0 \\
0 \\
0 \\
Vₛ
\end{bmatrix}
\]
CUSPICE Features

- It’s the NGSPICE version for CUDA Platforms
- It’s fully integrated in the NGSPICE framework
- It supports the following models:
  - Basic models
    - Resistor, Capacitor, Inductor, Voltage and Current Sources
  - Transistor models
    - MOSFET transistor (BSIM4v7)
- It requires at least a Fermi generation GPU
- Kepler architecture gives better performances
Model Evaluation in the GPU

- Linked lists are used to store models and instances
  - This is not good for the GPU
  - They have been converted into arrays
- Many branches (if-else) are related to fixed parameters
  - Temperature
  - Process
- Reorganize the code (slightly)
  - Minimize thread divergence
  - Maximize memory coalescence
Model Evaluation in the GPU

1. Allocate Arrays in the CPU and in the GPU
2. Convert the Linked list to Arrays in the CPU
3. Copy the Filled Arrays to the GPU
4. Start the Model Evaluation in the GPU
Model Evaluation in the GPU

1. Start the Model Evaluation in the GPU
2. Copy back the result arrays to the CPU
3. Update the Circuit Matrix and RHS

MODELS

INSTANCES ARRAY

ONE ARRAY FOR EACH PARAMETER

INSTANCES

instance ID 0 1 2
Model Evaluation in the GPU

- Every CUDA thread computes one instance
- **Coalesced access to the global memory is exploited**
- CUDA Streams have been used during the evaluation of p-type and n-type transistor (BSIM4v7)

![Diagram showing model evaluation process in CUDA](image)

- instance ID = blockIdx.x * blockDim.x + threadIdx.x
Basic Device Model Evaluation

![Graph showing speedup vs. number of instances of models.]

- **Resistor Netlist**: all resistors;
- **Capacitor Netlist**: half capacitors and half resistors;
- **Inductor Netlist**: half resistors, quarter capacitors and quarter inductors

*Resistor Netlist: all resistors; Capacitor Netlist: half capacitors and half resistors; Inductor Netlist: half resistors, quarter capacitors and quarter inductors

*NVIDIA C2070, ECC on
*Intel X5690 (6 Core™) @ 3.47GHz
BSIM4v7 Device Model Evaluation

ISCAS85 Benchmark Suite

- CPU (1 core)
- GPU

Time (ms)

6.67x

c432, c499, c880, c1355, c1908, c2670, c3540, c5315, c6288, c7552

*NVIDIA C2070, ECC on
*Intel X5690 (6 Core™) @ 3.47GHz
Circuit Matrix and RHS Update

After Model Evaluation in the GPU

For each Timestep {
    For each Newton-Raphson Iteration {
        CPU
        Input Parameters
        Model Evaluation
        KLU
        Circuit Matrix and RHS Update
        Output Parameters in Arrays
    }
}

GPU

These MUST be avoided
Circuit Matrix and RHS Update

After Model Evaluation in the GPU and moving the Circuit Matrix and RHS Update on the GPU

For each Timestep {
  For each Newton-Raphson Iteration {

  CPU

  Input Parameters

  KLU

  }

  GPU

  Model Evaluation

  Output Parameters in Arrays

  Circuit Matrix and RHS Update using ATOMICS

  Serial Operations in the GPU
  No guarantee of operation order
  No Coalesced Access

}
The entire Update Process has been changed completely, using a totally new approach, that we have called **Topology Matrix Method**.

The models still write their results in an array form.

The arrays are not copied back to the CPU anymore and the Circuit Matrix and RHS are **Created** (and not **Updated**) directly on the GPU.

There are 2 Topology Matrices (Circuit Matrix and RHS).
Every Topology Matrix is created upfront, during the Setup Phase, in the following way:

\[
\begin{bmatrix}
A_{11} \\
A_{21} \\
\vdots \\
A_{n1} \\
A_{nn}
\end{bmatrix}
\begin{bmatrix}
-1 & 0 & \cdots & 0 & \cdots & -1 \\
0 & 0 & \cdots & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\
1 & 0 & \cdots & -1 & \cdots & 1 \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\
0 & -1 & \cdots & 0 & \cdots & 0
\end{bmatrix}
\begin{bmatrix}
Value_1 \\
Value_2 \\
\vdots \\
Value_3 \\
Value_4
\end{bmatrix}
\]

Rows depend on the circuit
Columns are arbitrary

CSC Circuit Matrix or RHS

CSR Topology Matrix for the Circuit Matrix or the RHS

Unique Values from Device Models

Rows order has to follow the Topology Matrix columns order
The Topology Matrices are extremely sparse (just 7-8 elements per row); the ground node is not included.

\[
\begin{bmatrix}
A_{11} \\
A_{21} \\
\vdots \\
A_{n1} \\
A_{nn}
\end{bmatrix} =
\begin{bmatrix}
-1 & 0 & \cdots & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\
1 & 0 & \cdots & -1 & \cdots & 1 \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\
0 & -1 & \cdots & 0 & \cdots & 0
\end{bmatrix}
\times
\begin{bmatrix}
Value_1 \\
Value_2 \\
\vdots \\
Value_3 \\
Value_4
\end{bmatrix}
\]

KLU takes care of the Circuit Matrix

Just 1 bit per value is needed to code the Topology Matrix

For example, G is the only value that comes out from the Resistor Model

DEVtopology is called for every Device Model

A Position Vector is used to easily change the values position
In some circuits, LTE is still about the 10% of the total simulation time.

LTE is:
- A sequence of checks to determine a new time step
- The final new time step is the minimum of the previous ones

LTE needs to be parallelized as well.
**Total Simulation Time without Parsing and Setup Time**

![Chart showing simulation times for different benchmarks on CPU and GPUs.]

- **ISCAS85 Benchmark Suite**
- **GPU - C2070**
- **GPU - K20**

- **CPU**
- **GPU - C2070**
- **GPU - K20**

* NVIDIA C2070 and K20@600Mhz
* Intel X5690 (6 Core™) @ 3.47GHz
Total Model Evaluation Time (OP + Transient)

ISCAS85 Benchmark Suite

*NVIDIA C2070 and K20@600Mhz
*Intel X5690 (6 Core™) @ 3.47GHz
Total Model Evaluation Time in Transient Analysis

ISCAS85 Benchmark Suite

- CPU
- GPU - C2070
- GPU - K20

Time(s)

* NVIDIA C2070 and K20@600Mhz
* Intel X5690 (6 Core™) @ 3.47GHz
How to get CUSPICE

In order to get CUSPICE, you need to:

1) Clone NGSPICE GIT repository
2) Execute ‘git checkout CUSPICE’
3) Execute ‘./autogen.sh’
4) Execute ‘./configure -enable-cuspice …’
5) Execute ‘make’
6) Execute ‘make install’

You need to have NVCC installed (pick up the latest version to be sure it works)
In addition you need to have Autotools (and maybe TCL) installed
Conclusion

SPICE simulation has two most time consuming parts
- Device Model Evaluation
- Linear System Solution

Device Model Evaluation
- Speedup* up to 6.67x for BSIM4v7 MOSFET Model
- Speedup* up to 35x for Resistor Model

Entire Simulation
- Speedup* up to 3.74x OVERALL (without Parsing and Setup Time)
- Speedup* up to 8.89x in pure Transient Analysis

*speedup depends on the circuit
Thanks for your attention

QUESTIONS?