Parallel Diode Simulation

Kuang Chung Wang, Ravi Shivaraman, Karl Bo Lopker

June 13, 2012

Abstract

The use of simulations is essential in order to study the effects of composition uctuations when devices are scaled to deep submicron dimensions. A 1-D drift-diffusion device simulator has been developed to effectively simulate pseudomorphic p-n junction device on a distributed memory multiprocessor computer. The drift-diffusion equations are discretized using a finite difference method on an unstructured mesh. The obtained set of equations is solved in parallel using optimized linear algebra libraries. We have applied our simulator to a 1000 nm p-n junction device based on the GaN/InGaN interface.

1 Overview of Problem

Device simulators require large amounts of memory and high power CPUs because calculation times increase exponentially with the number of nodes in a simulation mesh. For example a common device is with the scale of $1\mu m^3$, when we want to approch the grid to the atomic level, the number of grid points become 10^12 points. In this work we present a device simulator which uses the finite difference method within the drift-diffusion approximation to semiconductor transport. Parallel algorithms are employed to speed up the whole simulation process. The simulator has been developed for shared-memory computers using a Multiple Instruction-Single Data strategy (MIMD) under the Single Program-Multiple Data paradigm (SPMD).

As shown in Fig.1, Drift diffusion is solved by Gummel iteration method where three steps are taken and iteratively for convergence. First is to solve the nonlinear Poisson with Newton method which is an iteration method. Second and third is to solve the continuity equation for electron and holes. Two $A * x = b$ solving are required with A being a sparse matrix. Another method is to use newton method for all three equation instead of one. The simultaneous way can converge to the solution faster when it is close to the real solution however diverge fast if the initial value is bad. Furthermore it will consume three times as much as memory.

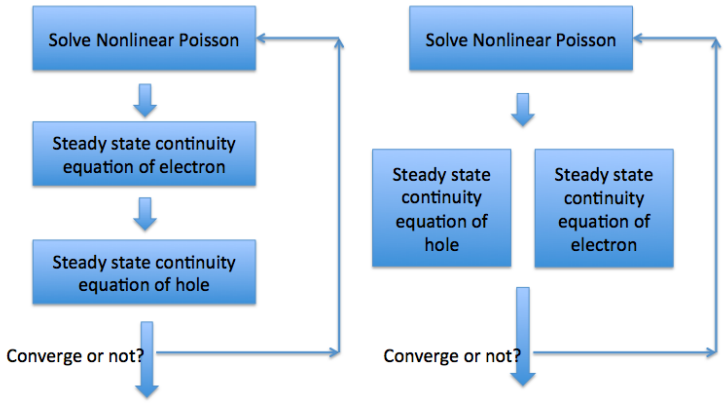

Figure 1: The illustration of the iteration and high level of parrellization.

2 Background

Semiconductors are materials that have properties in between normal conductors (materials that allow electric current to pass, and insulators. Semiconductors fall into two broad categories. First, there are intrinsic semiconductors. These are composed of only one kind of material, eg, Silicon. Extrinsic semiconductors are made of intrinsic semiconductors that have had other substances added to them to alter their properties. One may also dope the semiconductor material. Semiconductor materials are doped with impurities chosen to give the material special characteristics. One may want to add extra electrons or remove electrons. Doping atoms are chosen such that they differ in valence but have similar size to the intrinsic element. Thus individual intrinsic semiconductor atoms may be replaced with dopant atoms to form an extrinsic semiconductor. The binding energy of the outer electron added/removed by the impurity is weak. This is represented by placing the excess electrons just below the conduction band. The removal of electrons is represented by holes and these are placed just above the valence band. Thus very little energy is required to move these electrons into the conduction band or holes into the valence band. For example, an n-type (electron added/donor) extrinsic semiconductor operating at room temperature will have most of these "extra" electrons existing in the conduction band. Thus at normal operating temperature, $n \simeq N_d$.

Electron Mobility: When an electric field is applied to a semiconductor, the electrons experience a force and are accelerated in the opposite direction of the electric field. This acceleration is inhibited by what we term 'collisions'. When a collision occurs, the velocity of the electron drops to zero and it accelerates again. The average time between collisions is given by τ The effect is a constant drift velocity for an n-type semiconductor given by: v_n = -μ_n * E where E is the electric field and μ_n is the electron mobility. Also, $\mu_n = \frac{q*\tau}{m_e}$ Diffusivity: When a electron concentration gradient ∇n is present in the device, electrons

move from a region of higher chemical potential (higher concentration or higher (quasi) Fermi energy to normalize the potential across the device. This leads to the diffusion current and the corresponding Diffusivity is related to the electron mobility by the relation $D_n = \frac{\mu_n * k_B * T}{q}$.

These are thermodynamic quantities that describe the transport properties of charge carriers in the Drift Diffusion Solver.

3 Methodology

3.1 Nonlinear Poisson equation

3.1.1 Equations to be solved:

The equations in use are:

$$n = n_i exp(\frac{E_{fn}}{k_b * T}) exp(\frac{-qV}{k_b T}) \tag{1}$$

$$p = n_i exp(\frac{-E_{fn}}{k_b * T}) exp(\frac{qV}{k_b T}) \tag{2}$$

$$N_b = N_a - N_d; \tag{3}$$

$$\nabla^2 V = \frac{q}{\epsilon}(-n - p - N_b) \tag{4}$$

Eqn.1 and 2 desribe the density of carriers, which is related to the distance between fermi energy E_{fnp} and the intrinsic potential V. Carriers and the ionized dopants 3 contribute to the total charges in the system and can give rise to potential difference V according to Poisson equation, eqn.4.

3.1.2 Implementation

In this nonlienar poisson process, we are given E_{fn} and E_{fp} and seek for the V to balence the above four equantions. However, this becomes a nonlinear equation where V is embeded inside the exponetial term. Therefore we let $V = V_0 + dV$, assuming dV is small and use Taylor expansion to get eqn.8.

Therefore, each iteration we do the following:

$$n = n_i exp(\frac{E_{fn}}{k_b * T}) exp(\frac{-qV}{k_b T}) \tag{5}$$

$$p = n_i exp(\frac{-E_{fn}}{k_b * T}) exp(\frac{qV}{k_b T}) \tag{6}$$

$$N_b = N_a - N_d; \tag{7}$$

$$\{\nabla^2 + \frac{q^2 n_i}{\epsilon k_B T}(-n + p)\}\delta V = -\nabla^2 V + \frac{q}{\epsilon}(-n - p - N_b) \tag{8}$$

$$V = V + \delta V \tag{9}$$

where beside k_b, T, q, ϵ, all the others are vetors with size N.

The grid size in the system is decided by two times the Debye length:

$$dx = \sqrt{\frac{\epsilon k_b T}{q^2 N}} \tag{10}$$

This is becuase of eqn.8, where the laplacian is porportional to dx^{-2} and the second term proportional to debye lenght. For the ∇^2 to be significant, here comes this restriction.

V needs boundary condition. There are two kinds of boundary conditions. One is those boundaries in touch with the contact which we use fixed boundary condition. The other is one not touching which we use floating boundary condition. In our system,

$$x = [x_1 x_2 x_3 ... x_N] b = [b_1 b_2 b_3 ... b_N] \tag{11}$$

1. Floating boundary condition:

$$\nabla^2 V = b \tag{12}$$

$$x'(1) = x'(2) \rightarrow \frac{x_2 - x_0}{2h} = \frac{x_3 - x_1}{2h} \rightarrow x_0 = x_2 - x_3 + x_1 \tag{13}$$

$$x_0 - 2x_1 + x_2 = b_1 \tag{14}$$

$$x_2 - x_3 + x_1 - 2x_1 + x_2 = b_1 \tag{15}$$

$$2x_2 - x_3 - x_1 = b_1 \tag{16}$$

$$x'(N) = x'(N-1) \rightarrow \frac{x_{N+1} - x_{N-1}}{2h} = \frac{x_N - x_{N-2}}{2h} \tag{17}$$

$$\rightarrow x_{N+1} = x_{N-1} + x_N - x_{N-2} \tag{18}$$

$$x_{N+1} - 2x_N + x_{N-1} = b_N \tag{19}$$

$$x_{N-1} + x_N - x_{N-2} - 2x_N + x_{N-1} = b_N \tag{20}$$

$$-x_{N-2} + 2x_{N-1} - x_N = b_N \tag{21}$$

Therefor the matrix becomes:

$$\nabla^2 = \begin{bmatrix} -1 & 2 & -1 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 \\ & & & & & \\ 0 & 0 & ... & -1 & 2 & -1 \end{bmatrix} b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \\ \end{bmatrix},$$

2. Fixed boundary condition:

$$\nabla^2 V = b \tag{22}$$

$$x(0) = \sigma \tag{23}$$

Therefor the matrix becomes:

$$\nabla^2 = \begin{bmatrix} -2 & 1 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 \\ & & & & & \\ & & & & & \end{bmatrix} b = \begin{bmatrix} b_1 - b_0 \\ b_2 \\ b_3 \\ \\ \end{bmatrix},$$

3.1.3 Graphical Demonstration

We simulate a p-n junction with p-type with dopants $N_p = N_a = 1X10^{17}cm^{-3}$. The length of the device is 500 nm n-type and 500 p-type.The material parameter is Si with 1.1 eV of bandgap.

Figure 2 shows the convergence of the intrinsic voltage. The inital condition given is a straight line connecting the two boundary points. As the time progresses, we can see that the left side moves up and the right side move down. The corresponding carreir concentration is shown in Fig. 3, the carriers become more on either side of the junction with holes on left side and electron on the right side.

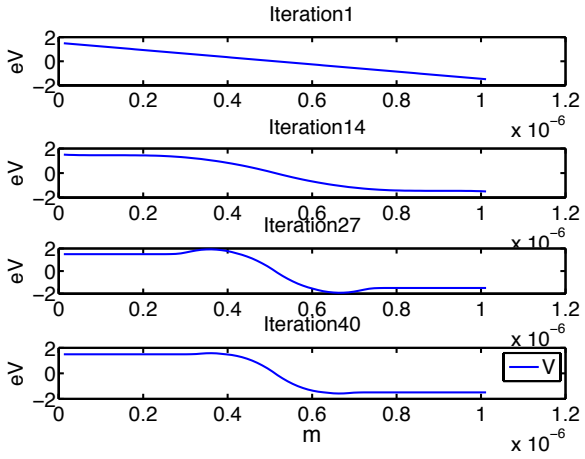

Figure 2: The convergence V.

To test multiple layers, a quantum well structure is insterted in between the n-type and p-type GaN. The quantum well is set to be InGaN with a $E_g = 1.6eV$. From Fig. 4, we can see a dip in the middle and in Fig. 5 we can see tow spikes which shows more carriers in the well.

3.2 Continuity Equation

The Poisson equation yields the potential profile and gives initial values for the concentrations. Under an application of a voltage bias, current flows in the device. The current in the device can be computed by taking the gradient of the Fermi energies. The second step computes the Quasi Fermi levels. Convergence has been achieved in calculation of the Quasi Fermi levels and the current density is calculated. The equations employed to perform these calculation and results produced are shown below.

The following tridiagonal systems are solved in 1-dimension for the electrons

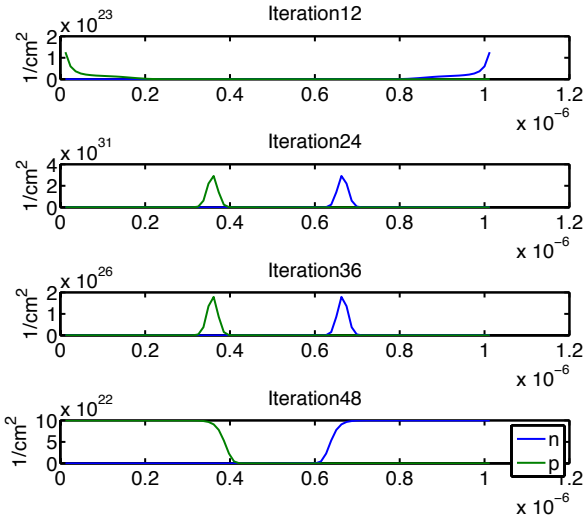

Figure 3: The convergence of n and p.

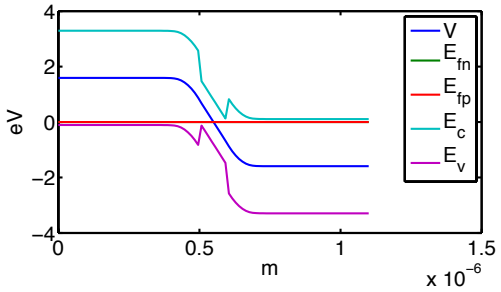

Figure 4: Band diagram for a qauntum well structure.

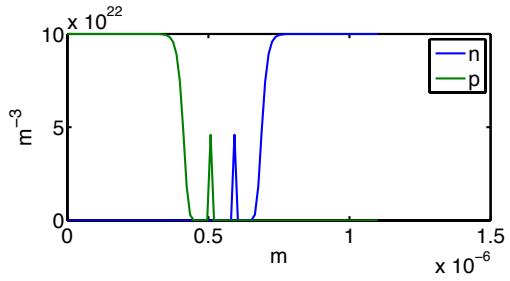

Figure 5: n and p for a quantum well structure.

and holes respectively:

$$a_n(i-\tfrac{1}{2})\Phi_n(i-1) - (a_n(i-\tfrac{1}{2}) + a_n(i+\tfrac{1}{2}))\Phi_n(i) + a_n(i+\tfrac{1}{2})\Phi_n(i+1) = \triangle x^2 U_n \quad (24)$$

$$a_p(i-\tfrac{1}{2})\Phi_p(i-1) - (a_p(i-\tfrac{1}{2}) + a_p(i+\tfrac{1}{2}))\Phi_p(i) + a_p(i+\tfrac{1}{2})\Phi_p(i+1) = \triangle x^2 U_p \quad (25)$$

$$(26)$$

Here U_n and U_p take into account generation-recombination terms. The coefficients a_n and a_p are given by

$$a_n = n_i \mu_n exp(\tfrac{-qV}{k_B T}) \quad (27)$$

$$a_p = -n_i \mu_p exp(\tfrac{qV}{k_B T}) \quad (28)$$

$$(29)$$

The interpolation is done carefully in-between the grid points. The variables Φ_n and Φ_p are called Slotboom variables and are respectively given by:

$$\Phi_n = exp(\tfrac{E_F n}{k_B T}) \quad (30)$$

$$\Phi_p = exp(\tfrac{-E_F p}{k_B T}) \quad (31)$$

$$(32)$$

FORWARD BIAS: The application of a positive voltage at the p-end leads to the holes in the P-type region and the electrons in the N-type region being pushed toward the junction. This reduces the width of the depletion zone. The positive charge applied to the P-type material repels the holes, while the negative charge applied to the N-type material repels the electrons. As electrons and holes are pushed toward the junction, the distance between them decreases. This lowers the barrier in potential. This is called forward bias. Only majority carriers (electrons in N-type material or holes in P-type) can flow through a semiconductor. With this in mind, consider the flow of electrons across the junction. The forward bias causes a force on the electrons pushing them from the N side toward the P side. With forward bias, the depletion region is narrow

enough that electrons can cross the junction and inject into the P-type material. However, they do not continue to flow through the P-type material indefinitely, because it is energetically favorable for them to recombine with holes. The average length an electron travels through the P-type material before recombining is called the diffusion length, and it is typically on the order of microns.

REVERSE BIAS: Because the p-type material is now connected to the negative terminal of the power supply, the 'holes' in the P-type material are pulled away from the junction, causing the width of the depletion zone to increase. Likewise, because the N-type region is connected to the positive terminal, the electrons will also be pulled away from the junction. Therefore, the depletion region widens, and does so increasingly with increasing reverse-bias voltage. This increases the voltage barrier causing a high resistance to the flow of charge carriers, thus allowing minimal electric current to cross the pn junction. The increase in resistance of the pn junction results in the junction behaving as an insulator.

Thus in order to describe the carrier concentration separately when the populations of the holes and electrons are displaced from equilibrium . This displacement could be caused by the application of an external electric potential, which injects electrons and holes, or by exposure to light of energy, which increases the density of both electrons and holes above their equilibrium values. The displacement from equilibrium is such that the carrier populations can no longer be described by a single Fermi level, and separate quasi-Fermi levels must be used for each charge carrier. When a disturbance from a thermal equilibrium situation occurs, the populations of holes and electrons change. If the disturbance is not too great or not changing too quickly, the populations of electrons and holes each relax to a state of quasi thermal equilibrium. Because the relaxation time for electrons within the conduction band is much lower than across the band gap, we can consider that the electrons are in thermal equilibrium in the conduction band. This is also applicable for holes in the valence band. In the case of electrons we can define a quasi Fermi level and temperature due to thermal equilibrium of electrons in conduction band and quasi Fermi level for holes similarly.

4 Implementation

The software was first prototyped in Matlab. This allowed us to make sure our modeling approach was correct before we parallelized, as is common practice. We chose Matlab because it has a better debugging interface than most languages which makes development quick. Once we proved our method to be mathematically correct we translated the code to C++ to parallelize. C++ is fairly bare-bones so we needed to decide which libraries to use. We didnt want to solve already solved linear algebra problems. Instead we wanted to use what we learned about analyzing parallel systems to make our program as efficient as possible. With that in mind, we chose a mix of linear algebra libraries to help us with that goal. First we chose the Armadillo C++ linear algebra library.

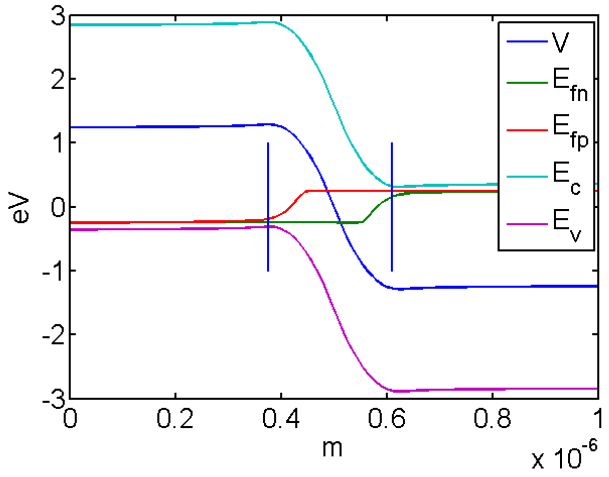

Figure 6: Forward Bias

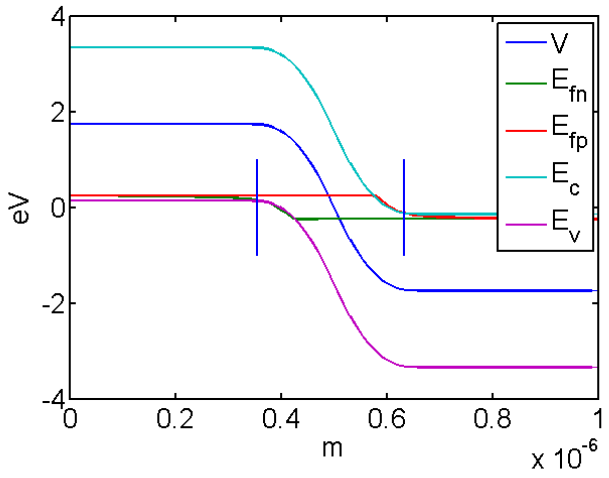

Figure 7: Reverse Bias

There are several C++ libraries like Armadillo, but we chose it for its Matlab like syntax which made the code conversion less painful (not painless though). Another advantage is that although Armadillo is a fully featured linear algebra library its core implementation can be swapped out for more optimized ones. We tested various other linear algebra libraries such as the reference versions of BLAS and LAPACK, GotoBlas, OpenBLAS and ATLAS. Finally, we decided to use ACML for our core linear algebra library. ACML is free, widely used and highly optimized. Although its made by AMD it works well on Intel architectures too. ACML contains all three levels of BLAS and the full LAPACK suite in one package. The only drawback is that its not opensource. Methodology

To test our simulation we set up a dedicated virtual machine on our development computer. Our development computer is an older Dell laptop with a first generation i5 Intel CPU. Although its quite a bit slower than a modern HPC cluster node, we needed root access to install the needed libraries. However, it does have eight cores which is perfect for demonstrating parallelism. The development computer uses VirtualBox 4.1.16 to create the virtualized environment. VirtualBox allows us to control exactly how many resources to give the virtual computer. We allotted it 4096MB of RAM and changed the number of cores for each test. The virtual machine is running Xubuntu 12.04 with Armadillo 3.2.2 and ACML 4.4.0-gfortran installed. We can now run our tests with the environment correctly configured. We ran eight different tests, one for each core of the machine. For every test we shut the virtual machine down completely, brought it back up and let it rest for one minute. This is to ensure the tests would not interfere with each other. Furthermore to control for various background processes we ran each test four times and picked the fastest overall run. Two parameters were used for testing. Each of the eight tests were done with a different number of cores available to it. During each test our program was run ten times as we increased the length of our simulated diode. We chose to vary the length parameter because it scales linearly with the problems size (N).

5 Result

In Fig. 8 see the effect a diodes length has on running time. As we increase the problem size linearly, we get a quadratic increase in time. This indicates that our algorithm is asymptotic to $O(n^2)$, which is the expected result since we have to solve several systems of equations. We also start to notice the effect parallelism has on our simulation. Using two processors almost halves the execution time. Adding more than two processors we can see some gain, but its difficult to tell from this graph.

Fig.9 shows the speedup for the longest diode we tested. This graph provides us with better clarity for whats happening as we increase the number of cores. At two cores the speedup is almost perfect at 1.909. Unfortunately as we add the eighth core speedup is only at 2.57. If we extrapolate our results the simulation speedup seems to approach 3., which hints that our parallelism is somewhere between 2.57 and 3. This result seems fair considering there are still large parts

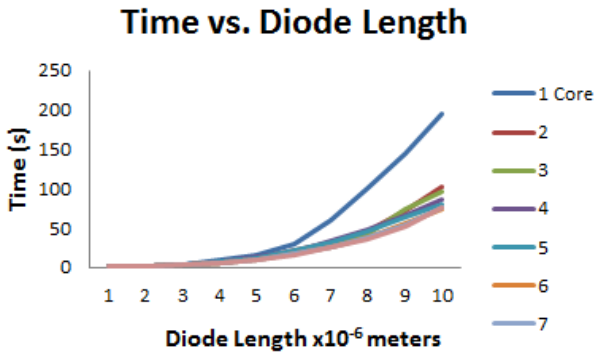

Figure 8: The calculation time comparison between different length of device.

of the code running sequentially. This behaviour is common for systems that either do not have enough memory bandwidth for processor communication or enough span in its algorithm. Since we are running on an older computer, our tests seem to exhibit both of these symptoms.

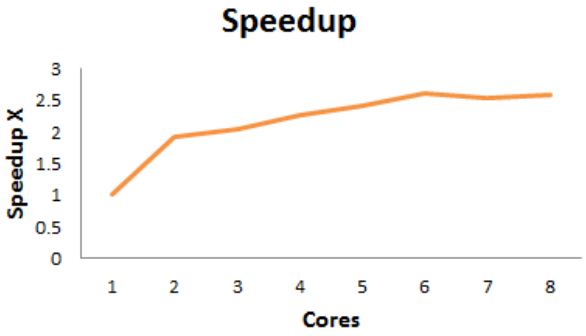

Figure 9: The speed up for vs differetn cores.

Lastly, Fig.10 shows the parallel efficiency for the same diode. Again here efficiency stays strong until we add more than 4 cores at which point it goes below half. Using 8 cores only gives us an efficiency of .322. This result confirms that our parallelism is likely around 3, but it also hints at a good suggestion for optimal processor count. If we want to stay above half efficiency the optimal number of cores is 4.

6 Discussion

From our results it is clear that the simulation has more work than span and cannot achieve perfect efficiency. Running our simulation on commodity hardware with relatively slow memory in a virtualized environment did not help the

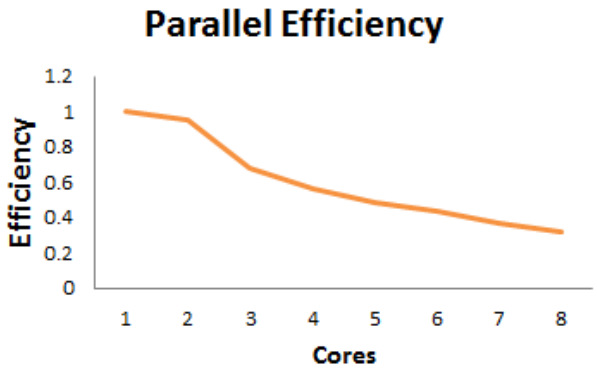

Figure 10: The efficiency versus differetn cores.

performance. However we can be assured that our results are the worst case and the simulation would only run better on performance hardware.

Although our results indicate our approach is not infinitely parallelizable, it simulates diodes well given a couple cores. Furthermore, we have showed that applying basic parallel principles to complex physical simulations can dramatically increase performance. Given that most every computer today has multiple cores, this is a fantastic realization.

In the future we would like to introduce more parallelism by using C++0X type threads. Threading would allow us to parallelize larger chunks of the simulation like calculating the n and p bands when calculating the Fermi levels. Also, we would like to try more linear algebra libraries like Intels MKL, which is not available for free.

Another future work would be to increase the dimensionality of the simulation, with the ultimate goal of efficiently simulating 3 dimensional diodes.

7 Conclusion

Creating a working sequential diode simulation is not a trivial task. Adding parallelism to it is even more difficult. In this paper we took a high level physical simulation plan and applied it in various code languages. We then optimized it using freely available linear algebra libraries. Then to show how effective our approach was we analyzed our programs parallelism and discussed what worked well and what could use improvement. We then discussed what future work could be done with these results. We would have liked to further optimize the programs parallelism, but understanding the physics and getting the code correct took significant effort and time.

8 References

1. Vasileska D., "Drift-Diffusion Model: Time-Dependent Simulations Sharfetter-Gummel Discretization", 1996,

2. Gummel, H.K., "A self-consistent iterative scheme for one-dimensional steady state transistor calculations", IEEE Transactions on Electron Devices, 1964,OCT; Vol. 11, No. 1, pages= 455 - 465, doi=10.1109/T-ED.1964.15364, ISSN="0018-9383",

3. Riordan M., Hoddeson L., "Crystal Fire: The invention of Transistor and the birth of the Information Age", 1998,ISBN= "0393318516".

4.Pierret R. F., "Semiconductor Device Fundamentals", 1996, Addison Wesley, April, ISBN= "0201543931