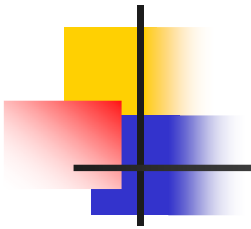
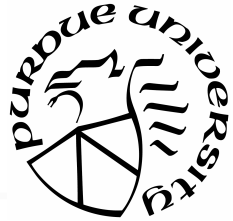
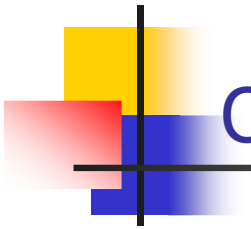




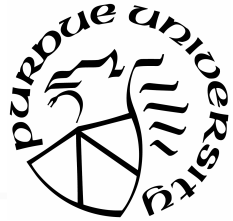
A guided tour of nanoMOS code and some tips on how to parallelize it.

Electron Devices at the Nano/Molecular Scale
May. 21-22, 2002
Summer School at UIUC

by: Sebastien Goasguen

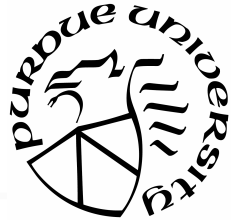


Outline



I. Spirit of nanoMOS
II. The 'big' and the 'small' routines
III. The post/pre-processing routines
IV. The computation routines
V. How to parallelize a Matlab application ?
VI. How/Where to parallelize nanoMOS ?
VII. Where do we go from here ?

Spirit of nanoMOS code



- nanoMOS is a set of Matlab routines (18)
- Our philosophy is that it is more important to focus on the physics than on the programming !
- Get the physics right first and then think about the implementation
- A scripting language like Matlab allows us to do that.
- Programming is easy and fast, a lot of pre-built routines are available.
- Therefore Matlab was the language of choice to implement new techniques like NEGF…and buttiker probes.

‘Big and small’ routines

- nanoMOS is approximately 4500 lines of code.
- The ‘big routines’ are:
  - ET.m … energy transport model
  - Charge.m … computes the charge density..transport models
  - Current.m … computes the current
  - Current_mat.m … computes the fermi_level of the buttiker probes (qdte model)
  - Main.m … self-explanatory
  - Parser.m … reads the input deck
  - Poisson.m … solves poisson’s equation
  - Readinput.m … pre-processing
  - Saveoutput.m … saves and plots output data
  - Schred.m … solves Schrodinger equation in the transverse direction

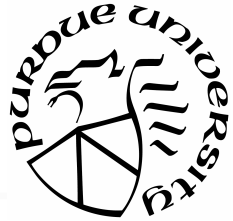
'Big and Small' routines

- The small routines are:
  - Dummy.m … Converts charge to a quasi-fermi level (for non-linear Poisson solver)
  - Anti_dummy.m … Converts quasi-fermi level to charge (for non-linear Poisson solver)
  - Dummy_prime.m …Differentiates dummy functions
  - Doping.m …Generates 2D doping profile
  - Fermi.m … Computes fermi integrals
  - Fprime.m …Differentiates fermi dirac integrals (for Poisson solver)
  - Integral.m …Computes the classical charge density (clbte model)
  - Nanomos.m … The main routine that begins simulations

Post/pre-processing routines

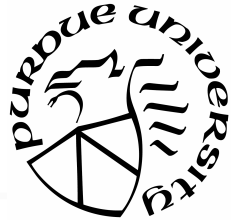
- Readinput.m
  - After reading the input deck using parser.m, the input values are stored with the right units and passed as global variables.
  - Some limits are set to reduce the computational cost: thin bodies (<2nm) simulations only use one subband. However, these limits can be overridden.
- Saveoutput.m
  - Plots and save all data specified in the input deck, can be easily extended to plot more graphs…recent extension: DOS and transmission coefficient
- Parser.m
  - Reads the input deck

Computation routines



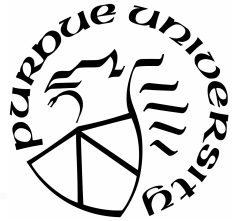
- The core of nanoMOS is in main.m
  - Variables are initialized
  - Doping profile is created (doping.m)
  - Pre-processing for Poisson solver (fprime.m)
  - Initial guess is computed using clbte model
  - Then solution is computed by iteration of calls to charge.m and Poisson.m…self-consistency
  - If I-V are required, it enters a loop over all bias points where same iteration scheme is used.

Computation routines



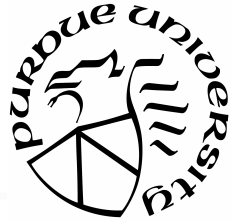
- Charge.m computes the charge density depending on the chosen model
  - If transport_model==….
    - Solve Schrodinger equation in the confinement direction for the subbands energies and distribution function (eigen energies, eigen function)
    - Applies transport model in the longitudinal direction
- Poisson.m solves Poisson's equation using the non-linear scheme
  - Converting the charge density to an equivalent fermi level
- Current.m computes the source and drain current for every different model
- Current_mat.m computes the fermi level of the buttiker probe to ensure zero scatterer current

Run nanoMOS with nanomos.m



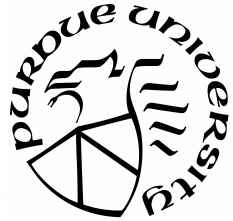
- This is the starting file…
  - Define global variables
  - Calls readinput to parse the input deck and assign values to variables
  - Calls main.m to start computation
  - Main.m can be timed at this point…
  - Calls saveoutput to post-process the data.

How to parallelize a Matlab application ?



- Code parallelization is done through message passing between processors.
- This is possible on standard super-computers or Linux Clusters
- Message passing is implemented by MPI (message passing interface) or PVM (parallel virtual machine)
- How to use MPI or PVM in Matlab ?
  - Use MPITB or PVMTB

PVMTB and MPITB for Matlab



- Javier Baldomero from the University of Granada/Spain has written PVMTB and MPITB.
- Available for Matlab 5.3 and Matlab 6.x
- Standard Unix systems and RedHat 7.*
- Mex functions bindings of MPI or PVM routines.
- You call MPI or PVM routines the same way you would in C/C++ or Fortran
- Easy to use and a very good tool to learn parallelization under Matlab environment.

Superman Linux Cluster

200 processors, 1.2 GHz
ATHLON, 1 GB RAM

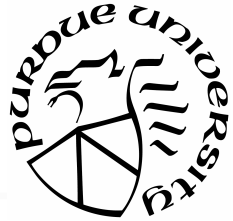
Benchmarked at 130 GFLOPS !

An example: Detailed Scattering model

- To validate the buttiker probe model used in nanoMOS, a detailed scattering model can be used.
- This is extremely computationally expensive.
- Non-self consistent simulation….4-5 days !!!!!!
- Self-consistent simulation….. 40 days !!!!!

- On 40 processors using PVM under Matlab…a self consistent simulation can be performed in 30 hours. A non self-consistent in 2-4 hours…

How/Where to parallelize nanoMOS ?



- The main goal is to distribute the computational load of a code among processors
- Identify parts of the code/algorithm than can be done simultaneously, for example:

  1- Different bias points or different set of bias points.

  2- Integration of functions

  3- Independent for loops (distribute the energy grid in NEGF simulations for ballistic case of elastic scattering)

  4-have a different processor for each valley and subband. Coupling between subbands complicates the problem but can be solved.

  5-Linear solvers…MatPar from JPL uses PVM and scalapack.

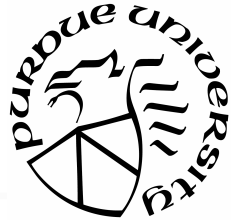
Where do we go from here ?

- Improvement is always possible and we welcome feedback/bug reports…
- Send bug reports to myself: sebgoa@purdue.edu
- Send general nanoMOS question to the Forums on the nanoHUB.
- nanoMOS being open source it is in the interest of everyone to share their enhancement to the code and maybe create a development group !
- We will continue adding more transport model…Density gradient, DESSIS…and add new features to model more various types of devices…Schottky, bulk, SOI, HBT…etc.

Where do we go from here ?

- A parallel version will be released soon, due to CPU requirements this won't be usable through the HUB immediately but only available for download.
- A a start two types of parallelization will be possible:
  - Sending different bias point to different processors
  - Distributing the energy grid in ballistic NEGF

Some useful links



- PVMTB home page (download,links)

  http://atc.ugr.es/javier-bin/pvmtb_eng
- PVM home page (links and tutorial)

  http://www.csm.ornl.gov/pvm
- PVM home page at netlib (source code)

  http://www.netlib.org/pvm3/
- MPITB home page

  http://atc.ugr.es/javier-bin/mpitb_eng
- Superman Web-page @ CELab

  http://ece.purdue.edu/celab