Nested parallelism in the drift-diffusion model for semiconductor devices

G. Gazzaniga[1], P. Lanucara[2], P. Pietra[1], S. Rovida[1], G. Sacchi[1]

gianna@imati.cnr.it lanucara@caspur.it pietra@imati.cnr.it rovida@imati.cnr.it gianni@imati.cnr.it

[1] *IMATI - CNR, v. Ferrata 1, Pavia, Italy*

[2] *CASPUR, v. Tizii 6b, Roma, Italy*

Outline

- Goals *(partially missed)*.
- Drift diffusion model - Iterative scheme.
- Test Case - Profiling.
- Architectures used.
- MPI parallelization - Results.
- Another approach.
- Hybrid parallelization - Results.
- Comments.

Goals

- Investigate the possibility to improve the performance of a sequential numerical FORTRAN code for modelling semiconductor devices.
- Avoid the expensive task of reengineering the original code.
- Mantain portability

Drift–Diffusion Model

- Poisson equation for the electrostatic potential ψ

$$-\lambda^2 \Delta\psi = p - n + C \ , \quad in \quad \Omega$$

C *doping profile* , λ *Debye length* (*small parameter*)

- continuity equations for the charge density p and n

$$div\underline{J}_p = 0 \ , \quad \underline{J}_p = -(\nabla p + p\nabla\psi) \ , \quad in \quad \Omega$$

$$div\underline{J}_n = 0 \ , \quad \underline{J}_n = \quad \nabla n - n\nabla\psi \ , \quad in \quad \Omega$$

$\underline{J}_p$, $\underline{J}_n$ *density current vectors*

- Dirichlet - Neumann boundary conditions.

Gummel Iterative Scheme

- For the numerical solution of the drift-diffusion model we choose a modification of the so called Gummel method, that can be regarded as an approximated Newton method.
- Such a scheme has the advantage that only three decoupled linear problems have to be solved at each step, instead of the original strictly–coupled system.

Continuation in λ

For small Debye lengths λ the Gummel iterations result very sensitive to the initial guess of p , n.

In order to guarantee the stability, a continuation in the parameter λ is performed:

- start with $\lambda \simeq 10^{-1}$ and any initial guess for p , n;
- decrease slowly λ using as initial values of p , n the solution just computed for the previous value of λ;
- stop if the physical Debye length has been reached.

Iterative Scheme -1

begin continuation in λ

begin Gummel iterations

Step 1 $-\lambda^2 \Delta \delta\psi + (p^k + n^k)\delta\psi = \lambda^2 \Delta \psi^k + p^k - n^k + C$

$$\psi^{k+1} = \psi^k + \delta\psi$$

Step 2 $div \underline{J}_p^{k+1} = 0 \ , \quad \underline{J}_p^{k+1} = -(\nabla p^{k+1} + p^{k+1} \nabla \psi^{k+1})$

Step 3 $div \underline{J}_n^{k+1} = 0 \ , \quad \underline{J}_n^{k+1} = \quad \nabla n^{k+1} - n^{k+1} \nabla \psi^{k+1}$

end Gummel iterations

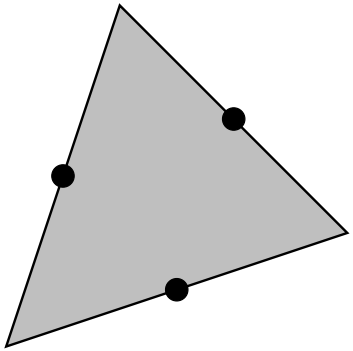
end continuation in λ

Iterative Scheme - 2

- *Step 1* is performed by means of P_1 non-conforming finite element method;
- *Step 2* and *Step 3* are solved via an exponential fitting mixed finite element scheme;
- the sparse linear systems coming from the discretization of the equations for ψ , p , n are solved using *GMRES* method (*NAG, SLAPACK, SPARSKIT2 Libraries*);

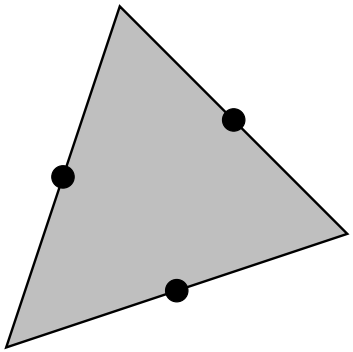
- the iterative scheme exhibits an intrinsic mathematical 2-way parallelism due to the independency of p and n equations in *Step 2* and *Step 3*.

Step 1



- P_1 non-conforming finite element method
- a *M-matrix* is obtained

Step 2 and Step 3



A well suited scheme to solve *Step 2* and *Step 3* should:

- well resolve boundary/internal layers;
- automatically adapt to pure diffusive and advection dominated regions;
- preserve the current;
- develop NO spurious oscillations.

exponential fitting mixed finite element scheme

Flow Chart

begin continuation in λ

  begin Gummel iteration

    Step 1 - solve equation in ψ

    Step 2 - solve equation in p

    Step 3 - solve equation in n

  end Gummel iteration

end continuation in λ

independent steps

- The code is written in FORTRAN, using BLAS and NAG Libraries.
- In the test case considered:
  - the external loop consists of ≈ 20 iterations in λ;
  - each λ requires $6 \div 10$ Gummel iterations.

NAG Fortran SMP Library - 1

- Why NAG Library ?
  - robustness
  - reliability
  - accuracy
- NAG SMP Library:
  - developed on OpenMP standard improving portability;
  - available on a broad range of platforms including Windows NT, SGI 6, Sun Solaris, Compaq Alpha ...

http://www.nag.co.uk/numeric/fl/FSdescription.asp

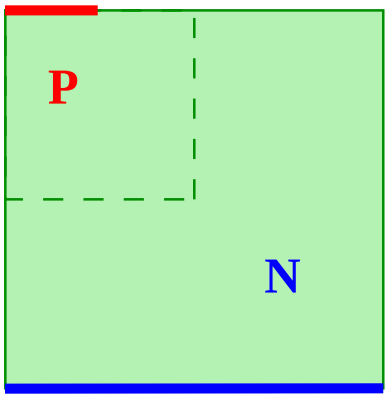
http://www..nag.co.uk/numeric/FL/manual19/html/genint/news_fs20.html

NAG Fortran SMP Library - 2

- From the point of view of performance and *scalability* the SMP-NAG Library consists in three sets of routines:
  1. *Tuned routines*: those that have been specially tuned and which therefore exibith near optimal performance and scalability;
  2. *Enhanced routines*: those that call one or more of the previous tuned routines;
  3. *Additional routines*: some of these perform BLAS operations as part of their algorithm and so benefit from the performance and scalability of the particular vendor BLAS.
- The routines used
  - *F11DAF* implements incomplete *LU* factorization
  - *F11DCF* implements *RGMRES, CGS, BiCGSTAB* methods, using the pre-conditioner computed by *F11DAF*

  belong just to the set 3.

Test Case - pn-Diode



Physical parameters

q	elementary charge	10^{-19}	$A\ sec$
ε	permittivity constant	10^{-12}	$A\ sec\ V^{-1}\ cm^{-1}$
U_T	thermal potential	0.025	V
$\overline{c}$	doping scale factor	10^{+16}	cm^{-3}
L	length of semiconductor device	10^{-3}	cm

$$\lambda^2 = \frac{\varepsilon\ U_T}{q\ \overline{c}\ L^2} = 2.5 \cdot 10^{-5} \quad squared\ Debye\ length$$

Different non uniform triangular meshes are used

test case	*xsmall*	*small*	*medium*	*large*	*xlarge*
number of elements	876	3504	7884	14016	21900
degrees of freedom (dof)	1354	5336	11946	21184	33050

Profiling

- The iterative scheme exhibits an intrinsic 2−way parallelism, due to the independency of *Step 2* and *Step 3*.
- Moreover the profiling of the serial code points out that *Step 2* and *Step 3* are just the most time consuming parts of the procedure, as shown by the cumulative times (*sec*) measured for the two smallest test cases.

SUN Entreprise 4500				
test case	*Step 1 time*	*Step 2 time*	*Step 3 time*	*execution time*
xsmall	21	72	86	183
small	102	674	939	1732

- The natural approach to the parallelization of the procedure is to distribute the computation of *Step 2* and *Step 3* among different processors.
- Due to the presence of a serial portion of the code (*Step 1 ...*) and to the imbalance between *Step 2* and *Step 3*, the expected *speed-up* is about $1.6 \div 1.7$.

Parallelization of the code

- OpenMP parallel programming model could be used, without any expensive re-engineering of the software, adding the *parallel sections* directive to the original code.

  The key factors of OpenMP are both the portability and the good efficiency on most shared memory platforms and compilers.
- We prefer another strategy of parallelization based on MPI message passing paradigm, to manage the interprocessor communication, in order to guarantee the portability on distributed memory architectures.

Architectures - 1

IBM SP4 node

- Power4 1.3GHz processor
- 8 CPU
- 16 GBytes RAM

- internode network: SP Switch 2

- ESSL-SMP, NAG-SMP Libraries
- OpenMP, MPI

CINECA, Bologna http://www.cineca.it

Architectures - 2

HP AlphaServer SC45 node

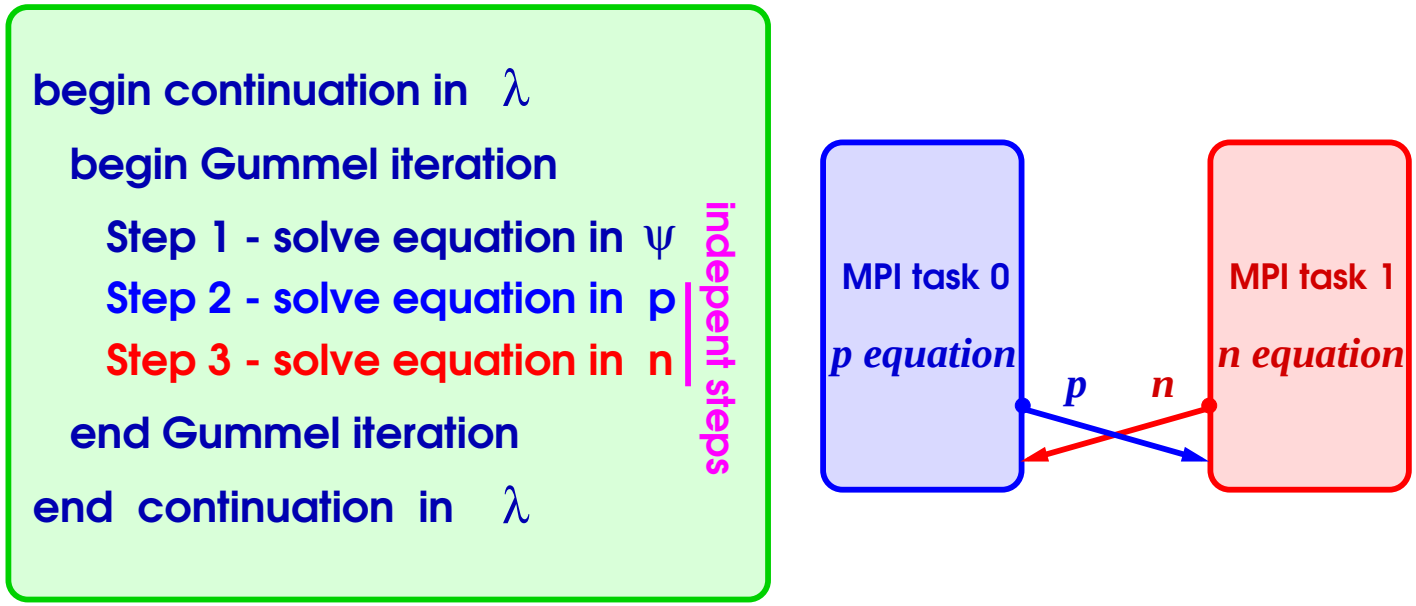
- Alpha-EV68 1.25GHz processor
- 4 CPU
- 16 GBytes RAM

- internode network: switch Quadrics Supercomputer Worlds

- CXMLP, NAG-SMP Libraries
- OpenMP, MPI

CASPUR , Roma http://www.caspur.it

MPI parallellelization



- Communications consist in the exchange between *task 0* and *task 1*, at each Gummel iteration, of the vector solutions p and n.
- The dimension of p and n vectors is exactly the number of degrees of freedom of the problem.

MPI Experiments - 1

- The table shows the execution time in *sec* for the largest test cases.
- The serial job is compared with the parallel job, running the two MPI-tasks on the same node (*MPI intranode*) and on different nodes (*MPI internode*).

IBM SP4			
	medium	*large*	*xlarge*
serial	584	1271	3175
MPI intranode	363 1.61	766 1.66	1918 1.66
MPI internode	368 1.59	759 1.67	1922 1.65

- The values of the *speed-up*, in blue, are the best one can achieve, according to the presence of a serial portion of the code and to the imbalance between the parallel sections.
- Intranode and internode runs give essentially the same performance due to the high speed internal network.

MPI Experiments - 2

- The table shows the execution time in *sec* for the largest test cases.
- The serial job is compared with the parallel job, running the two MPI-tasks on the same node (*MPI intranode*) and on different nodes (*MPI internode*).

HP SC45			
	medium	*large*	*xlarge*
serial	872	1878	4550
MPI intranode	568 1.54	1157 1.62	2857 1.59
MPI internode	780 1.12	1603 1.17	3448 1.32

- The values of the *speed-up*, in blue, for the intranode experiments are still satisfactory, according to the presence of a serial portion of the code and to the imbalance between the parallel sections.
- Internode runs exhibit a significative loss of performance.

MPI Conclusions

- Good performance at an acceptable programming cost.
- Deep knowledge of the logical structure of the numerical algorithm is not required.
- Issue of portability is guaranteed.

- Further improvement: multilevel parallelism.

A different parallelization approach

- *Step 1*, *Step 2* and *Step 3* require the solution of large sparse linear systems, carried out by means of a preconditioned *GMRES* solver.
- A further profiling of the code shows that for each *Step* most ($\approx 80\%$) of the time is just spent in the solver.

- Hint: use a multithreaded version of *GMRES*.

NAG SMP Results

- The original serial code has been used. Parallelism is achieved only by means of the multithreaded version of the *GMRES* solver.
- The experiments have been carried out for the largest test cases using NAG SMP Library release 2, varying the number of threads (*th*).

IBM SP4				HP SC45			
th	*medium*	*large*	*xlarge*	*th*	*medium*	*large*	*xlarge*
1	584	1271	3175	1	872	1879	4550
2	528 1.11	1125 1.13	2966 1.07	2	598 1.46	1295 1.45	3111 1.46
3	505 1.16	1066 1.19	2734 1.16	3	503 1.73	1082 1.74	2541 1.79
4	493 1.18	1040 1.22	2776 1.14	4	472 1.85	997 1.88	2499 1.82

- A satisfactory gain is obtained on HP platform, using few threads (2 or 3), taking into account the structure of the *GMRES* solver.

- On IBM-SP4 multithreading gives worst results.

Under Investigations

Why the multithreading gives worst performance on IBM-SP4 ?

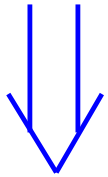
possible problems :

- The version of the linked NAG-SMP Library not optimized for the SP4 architecture.
- The implementation of the SMP version of ESSL included in the custom NAG solver.

I need an help !

Nested Parallelism

Results obtained on the HP architecture encourage to combine explicit message passing and shared memory parallelism.



Nested parallelism:

- MPI on the higher level;
- shared memory parallelism (NAG-SMP Library) on the lower.

Hybrid Experiments - 1

- The table shows the execution time in *sec* for the largest test cases.
- The two MPI tasks have been runned on the same node (*MPI intranode*) and on different nodes (*MPI internode*),increasing the number of threads used for the solver.

IBM SP4 - 2 MPI tasks						
	medium		*large*		*xlarge*	
th	*intranode*	*internode*	*intranode*	*internode*	*intranode*	*internode*
1	363 1.61	368 1.59	766 1.66	759 1.67	1918 1.66	1922 1.65
2	368 1.59	366 1.60	726 1.75	721 1.76	1895 1.68	1848 1.72
3	356 1.64	355 1.65	698 1.82	707 1.80	1735 1.83	1690 1.88
4	349 1.67	350 1.67	680 1.87	696 1.82	1740 1.83	1710 1.86

- As multithreading doesn't work, hybrid parallelism gives obviously worst performance.

Hybrid Experiments - 2

- The table shows the execution time in *sec* for the largest test cases.
- The two MPI tasks have been runned on the same node (*MPI intranode*) and on different nodes (*MPI internode*),increasing the number of threads used for the solver.

HP SC45 2 MPI tasks						
	medium		*large*		*xlarge*	
th	*intranode*	*internode*	*intranode*	*internode*	*intranode*	*internode*
1	568 1.54	780 1.12	1157 1.62	1603 1.17	2857 1.59	3448 1.32
2	395 2.21	609 1.43	833 2.25	1191 1.58	1996 2.28	2562 1.78
3	-	548 1.59	-	1081 1.74	-	2255 2.02
4	-	525 1.66	-	1047 1.79	-	2148 2.12

- Due to the configuration of the HP node, intranode experiments can be carried out with two threads at most.
- The *speed up*, in blue, are satisfactory for all the intranode runs.

Final comments

- Nested parallelism is an important topic;
- The mixed model MPI+OpenMP is the way to portability and efficiency on distributed shared memory systems

  (I don't know of any vendors that support nested parallelism for OpenMP);
- Further much could be gained using a more scalable multithreaded solver routine.

Acknowledgments

We wish to thank C. Calonaci and G. Erbacci of the CINECA (Bologna, Italy) for their assistance and for allowing us to use the IBM SP4 computing facilities.