Università di Pisa

A software platform for nanoscale device simulation and visualization

Marek Gayer
Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), Trondheim
Giuseppe Iannaccone
Dipartimento di Ingegneria dell'Informazione: Elettronica, Informatica, Telecomunicazioni, Università di Pisa

M.Gayer, G. Iannaccone, *A software platform for nanoscale device simulation and visualization* , Advances in Computational Tools for Engineering Applications, 2009. International Conference on ACTEA '09, pp. 432 – 437(2009).

A Software Platform for Nanoscale Device Simulation and Visualization

Marek Gayer and Giuseppe Iannaccone

Abstract— **NanoFEM platform is a new research environment based on the finite element method (FEM) for Technology CAD (TCAD) simulation and visualization of nanoscale devices, such as MOSFET transistors. The simulation in NanoFEM platform is based on solving partial differential equations corresponding to physical processes in modelled devices. A user or developer can provide these equations in a variational form format, and can define solver modules based on a FEM library with ability of automatic generation of finite elements and finite element forms. Solver modules can define fields for simulation and visualization and boundary conditions. Simple boundary conditions and material properties can also be specified directly in the graphical user interface. Geometry for the solved case can be defined either in graphical user interface or using Python scripting. Quality tetrahedral meshes necessary for FEM simulations are generated automatically. Visualization and post-processing is available in graphical user interface. We present some of related major existing solutions, namely open source geometry editors, mesh generators, computation libraries and visualization tools for FEM. We discuss major software components of the NanoFEM platform, i.e. Salome Platform and DOLFIN/FEniCS. We present an example simulation and visualization in NanoFEM platform. This is a simulation of the Poisson's equation on a 3D structure consisting of several geometry groups and materials forming a FinFET transistor with a mesh consisting of hundreds of thousands tetrahedrons. Because NanoFEM platform consists almost entirely of open source software components, others could eventually build similar solutions including, but not limited to TCAD device simulations after reading and reviewing the NanoFEM platform design and components.**

I. INTRODUCTION

A. Motivation

Semiconductor Technology CAD (TCAD)[1] has reached such a level of complexity that only extremely specialized engineers and physicists can use it to extract information relevant for technology development. Such consideration suggests that the typical business model of TCAD, based on software companies that develop the codes and sell licenses and support, has become inefficient and very costly for customers. In the medium term, the situation can worsen, since codes will become largely more complex, as devices enter the nanometer scale.

We have decided to design and implement a new scientific research platform for providing 3D modelling environment for Technology CAD (TCAD) simulations of nanoscale devices, such as MOSFET[2] transistors, based on the finite element method (FEM)[3]. However, our proposal might be interesting for other researchers, who effort to design and implement general simulation frameworks and projects based on the finite element method.

B. Finite Element Method

The finite element method is currently one of most often used techniques for numerical solutions of partial differential equations on complex modelled domains. Finite element method has become a defacto industry standard for solving wide variety of multi-disciplinary engineering problems. There are many applications using this method, such as solid mechanics, fluid mechanics, heat transfer, acoustics, electromagnetics and computational fluid dynamics.

For numerical computation using the finite element method on general 3D modelled objects, we need to provide a suitable mesh representation of the continuum. This means to find a suitable discretization of continuous domain to simple volume cell elements (e.g. triangles, tetrahedrons). In the cells, partial differential equations (PDE's) can be replaced by systems of non-linear algebraic equations. Using the finite element basis, discrete algebraic systems are assemblied into sparse matrices and then solved. Computed characteristics are determined in the nodes of the elements.

Computation of general partial differential equations using the finite element method is rather complex to design and implement, and solid understanding of mathematics, numerical methods and computer engineering is required for designing and implementing new performance optimal solutions of real problems based on this method.

One of important advantages of FEM is availability of software tools, pre- and post-processors as well as software components based on this method. This allows for instance to automatically generate meshes on arbitrary structures, and there are some good open source meshers for FEM. The wide availability of such tools suggests that the finite element method is in practice considerably more often used then similar methods, such as e.g. finite volume method.

II. RELATED SOLUTIONS

There are available several commercial and academic solutions and tools for TCAD simulation and modelling including: Crosslight Software (APSYS, LASTIP and PICS3D)[4];

This work was carried out during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme.

M. Gayer is with Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), Bragstads plass 2d, N-7491 Trondheim, Norway marek.gayer@itk.ntnu.no

G. Iannaccone is with Information Engineering Department, University of Pisa, Via Caruso 16, I-56126 Pisa, Italy giuseppe.iannaccone@iet.unipi.it

[1] http://en.wikipedia.org/wiki/Technology_CAD

[2] http://en.wikipedia.org/wiki/MOSFET

[3] http://en.wikipedia.org/wiki/Finite_element_method

[4] http://www.crosslight.com/Product_Overview/prod_overv.html

978-1-4244-3834-1/09/$25.00 © 2009 IEEE

ENEXSS Integrated 3D TCAD system[5]; gTs by Global TCAD Solutions [6]; SEQUOIA Design Systems - Device Designer[7]; Siborg - MicroTec: Semiconductor Process and Device Simulator[8]; Silvaco tools (Atlas, S-Pisces device simulator with DevEdit, DeckBuild, TonyPlot and TonyPlot3D)[9]; Synopsys Sentaurus Device: An advanced multidimensional (1D/2D/3D) device simulator[10]; TiberCAD[11] [1].

One of disadvantages of these tools is that users cannot develop their own code for new methods of device simulations, which is important for research and innovation.

There currently exist some open source or freely available tools for device simulation (sometimes only for non-commercial usage or provided without source code): Archimedes (2D Quantum Monte Carlo simulator for semiconductor devices)[12]; FLOODS/FLOOPS[13]; General-purpose Semiconductor Simulator (2D)[14]; NANOTCAD[15]; Nemo 3-D[16]; nextnano - Software for the simulation of electronic and optoelectronic semiconductor nanodevices and materials[17]. Pre-processing and post-processing for these free tools is limited.

III. CHOOSING FROM AVAILABLE FREE COMPONENTS

A. Finite Element Meshers

It is very complex to create a code for generation of 3D finite element mesh on arbitrary structures and therefore it is better to rely on existing meshers. During analysis of our NanoFEM platform project we took into consideration various open source meshers libraries. From these, NETGEN [2] and TetGen [3] seemed to be advanced in terms of stability, generality, quality, previous usage in other scientific projects and also suitability for our purpose.

B. Finite Element Simulation Environments

Some freely available open source environments for general finite element method analysis provide collection of components for tasks such as geometry modelling, meshing, visualization, common data structures, sometimes also an extensible framework for simulation modules and finite element solvers.

Gmsh [4] has some interesting features, such as integration of NETGEN and TetGen, geometry editor (although very basic and with only limited interactive features) and post-processing. There is support for different regions (using physical volumes). The user interface seem to be very non-standard and rather inconvenient. Gmsh is using a text format of mesh and data fields, for which it is easy to write a parser.

Calculix[18] is a 3D structural finite element analysis application. A graphical user interface seem to be less advanced than in Gmsh or even Salome Platform. It comes with solver CCX and pre- and post- processor CGX. It solves wide variety of mechanical, thermal, coupled thermomechanical, contact and field problems.

Salome Platform[19] is a very advanced finite element environment. Geometry editor is much more advanced than the one provided with Gmsh, and there is Python functionality for this editor (as well as for practically all other operations available through graphical user interface of Salome). Visualization and post-processing is much more powerful than in case of Gmsh, there exists a well documented C++ library for working with mesh and fields. There is a documentation on integration of own C++ and Python components into the Salome Platform. CORBA[20] functionality allows to run Salome components on remote servers, linking and integration of various simulation modules and control of simulation flow. Components can define their own user interface. With Salome Platform, we have a much more advanced and powerful simulation environment, that we would have with other solutions and components that we have tested. Salome Platform is available under the free LGPL license.

ORCAN [5] is a software with some features similar to Salome Platform, namely in component based architecture and set of modules for geometry modelling, meshing, visualization of results and automatic generation of user interface for components. There are also linear algebra solvers and material database. However, it is not as complete and mature as Salome Platform and is no longer actively maintained or developed since 2005.

We have decided to select Salome Platform, version 3.2.6 for geometry modelling, meshing, visualization of results and integration environment for our simulation platform.

C. Finite Element Solvers

We also decided to make selection of a suitable free finite element solver, that would make implementing equations running on semiconductor devices easier and faster, then if we would implement a new solver by ourselves.

We have reviewed many software solutions, libraries and projects, which address solving partial differential equations using the finite element method. From tens of candidates the most interesting for us included: FEniCS/DOLFIN[21] [6], libMesh[22] [7], Getfem++[23] , Rheolef[24], OOFEM.org[25] and OFELI[26].

[5] http://www.mizuho-ir.co.jp/english/solution/enexss/memory.html
[6] http://www.globaltcadsolutions.com/
[7] http://www.sequoiadesignsystems.com/products.html
[8] http://www.siborg.ca/microtec.html
[9] http://www.silvaco.com/
[10] http://www.synopsys.com/Tools/TCAD/Pages/default.aspx
[11] http://www.tibercad.org
[12] http://www.gnu.org/software/archimedes/
[13] http://www.flooxs.tec.ufl.edu/
[14] http://gss-tcad.sourceforge.net/
[15] http://monteverdi.iet.unipi.it/~fiori/ViDES/ViDES.html
[16] http://cobweb.ecn.purdue.edu/~gekco/nemo3D/
[17] http://www.nextnano.de/
[18] http://www.calculix.de/
[19] http://www.salome-platform.org/
[20] http://en.wikipedia.org/wiki/CORBA
[21] http://www.fenics.org/
[22] http://libmesh.sourceforge.net/
[23] http://home.gna.org/getfem/
[24] http://ljk.imag.fr/membres/Pierre.Saramito/rheolef/
[25] http://www.oofem.org/en/oofem.html
[26] http://www.ofeli.net/

We applied various evaluation criteria's, such as ongoing development, quality of documentation, fitness for devices simulations, number of developers, community around the project, available features, generality, easiness of use, extendibility and references from other scientific projects using it. FEniCS/DOLFIN (LGPL license) in our evaluations and comparisons was found to be the most suitable. Therefore, for the selection of an optimal finite element method based solver library, we selected DOLFIN/FEniCS version 0.7.3.

IV. COMPONENTS OF NANOFEM PLATFORM

A. Salome Platform

For pre-processing, i.e. namely defining the geometry of physical device and meshing we use Salome Platform. This way, we can use an advanced geometry editor and automatic finite element mesher on arbitrary 3D structures defined in the geometry editor. Geometry for the solved case can be defined either in a graphical user interface or using Python scripts. Quality tetrahedral meshes necessary for FEM simulations are created automatically, and we can define separate geometry and mesh groups. For visualization and post-processing, we can use advanced visualization features with 2D and 3D plots and graphs.

Salome Platform provides also functionality for exchanging data between codes and solvers in memory, CORBA to allow communication of modules on remote servers, and persistent data storage of all data based upon HDF format[27] (developed by Boeing and NASA in the area of Computational Fluid Dynamics) and MED format[28].

Because of Salome implementation and because we use nanostructures with very small dimensions, we had to multiply all the coordinates of modelled geometry by the scaling factor 10^9. We also count with this scale factor in our modules based on DOLFIN and Salome libraries. If we would use a smaller scaling factor (e.g. 10^3 or 10^6), we would obtain errors namely when generating a mesh or during visualization in Salome Platform.

By using Salome Platform, we save a lot of work and effort, which otherwise would be necessary to design and implement the mentioned features.

B. Finite Element Method Library DOLFIN

In order to implement equations, which would run on a device with finite element method, we are using a finite element library DOLFIN. DOLFIN is a C++ interface and library of FEniCS. This library offers many advantages and makes coding both easily and powerful. It supports iterative and direct solvers (LU decomposition[29], Krylov solver[30]) for sparse matrices[31], uses high-performance linear algebra libraries PETSc (with MPI)[32] and uBLAS[33] for solving systems of linear and nonlinear equations.

DOLFIN supports general families of finite elements, including arbitrary order continuous and discontinuous Lagrange finite elements, BDM elements, RT elements, BDFM elements, Nedelec elements and Crouzeix-Raviart elements. In DOLFIN, we use a C++ interface for communicating low level routines (functions) for evaluating and assembling finite element variational forms (UFC). DOLFIN is written in C++ and is using simple, intuitive and well structured object interface. DOLFIN is using various Krylov methods (the conjugate gradient method, the GMRES method, the stabilized biconjugate gradient squared method) and preconditioners[34] (no preconditioning, simple Jacobi preconditioning, successive over-relaxation, incomplete LU factorization, incomplete Cholesky factorization, algebraic multigrid).

C. Automation of the Finite Element Method

One of important advantages of basing our computational module on DOLFIN, is that we can automate the finite element method [8]. Detailed knowledge of the finite element method is not necessary to use and develop using DOLFIN.

This solves an important desire when designing simulations for the devices: possibility for easy testing of various provided equations, without necessity for manually programming the finite elements. A required equation has to be coded in the variational form format (with bilinear and linear components a and L) and then coded using a code with Python language syntax. A compiler utility FFC [9] is then used to generate corresponding C++ class. From these bi/linear form of equations in variational form, FFC generates UFC forms for evaluating and assembling finite element variational forms [10]. FFC makes automation of discretization, i.e. the automatic translation of a differential equation into a discrete system of equations. This saves effort, which otherwise would have to be put in when manually programming finite elements forms when new equations would be provided.

Standard variational formulation of partial differential equations is used: Find $u \epsilon V$ such that $a(v,u) = L(v) \quad \forall v \epsilon \hat{V}$, where $a : \hat{V} \times V \rightarrow \mathbb{R}$ is a bilinear form, $L : \hat{V} \rightarrow \mathbb{R}$ is a linear form and $(\hat{V}, V)$ is a pair of suitable function spaces [9].

For the Poisson's equation (1), where ε is permittivity, u is electric potential and f is source function, defined in the domain Ω, the corresponding bilinear form is (2) and linear form is (3), where g is the Neumann boundary condition.

$$\nabla(\varepsilon \nabla u) = f \tag{1}$$

$$a(v,u) = \int_{\Omega} (\nabla v \cdot \nabla u)\, \varepsilon \, dx \tag{2}$$

$$L(v) = \int_{\Omega} v f \, dx + \int_{\partial\Omega} \varepsilon v g \, ds \tag{3}$$

The corresponding Python style code for FFC compiler of this equation is on Fig. 1.

[27] http://hdf.ncsa.uiuc.edu/
[28] http://www.code-aster.org/outils/med/
[29] http://en.wikipedia.org/wiki/LU_decomposition
[30] http://en.wikipedia.org/wiki/Krylov_subspace
[31] http://en.wikipedia.org/wiki/Sparse_matrix
[32] http://www.mcs.anl.gov/petsc/petsc-as/
[33] http://www.boost.org/doc/libs/release/libs/numeric/ublas/doc/index.htm
[34] http://en.wikipedia.org/wiki/Preconditioner

```
# Compile this form with FFC:
# ffc -l DOLFIN PoissonEps.form
#
element = FiniteElement("Lagrange", "tetrahedron", 1)
v = TestFunction(element)
u = TrialFunction(element)
f = Function(element)
g = Function(element)
eps = Function(element)
a = dot(grad(v), grad(u))*eps*dx
L = v*f*dx + eps*v*g*ds
```

Fig. 1. The variational form of the Poisson's equation for the FFC compiler

Compiling this form using FFC results in a generated C++ header file with about 5000 lines, which can be used in a C++ code, such as in our computational module. More details can be found in FEniCS/DOLFIN manuals [11], [12].

D. Salome Platform - FEniCS/DOLFIN bridge MeshAPI

We created a library called MeshAPI, which is dedicated to establish connection and cooperation between Salome and DOLFIN and which provides their functionality for computational modules. The provided features include work with mesh, fields, material database, linear and nonlinear PDE solvers, selection of Krylov methods and preconditioners, boundary conditions and pre-coded solvers for equations. MeshAPI also provides C++ wrappers for connection and integration into Salome Platform and other functionality.

MeshAPI allows reading mesh from MED files to memory (using MEDMEM API), processing mesh coordinates and connectivities, working with groups of mesh (which can be defined in Salome editor) and passing this information to DOLFIN (to build mesh in memory). It provides some core fields (such as source, flux, potential) meant to be used in general device simulations. Custom fields can be defined as well. The field data structure is based on Salome FIELD data type, and we provide overloaded array access operators for easy access of the items in the code, as well as support for retrieving data from DOLFIN functions to MEDMEM and saving all fields to Salome MED files.

1) Material Database: Material database is specified in a XML file. Only the necessary parameters must be specified. Transforming of a XML file to C++ structures in MeshAPI is based on a SAX parser[35] using Libxml2[36]. For an example of XML material database, see Fig. 2.

2) Materials and Basic Boundary Conditions: Salome Platform promises DATA module, which would address necessity of assigning materials and other characteristics such as simple boundary conditions to a mesh. However, this module is not available in Salome 3.2.6, so we have decided to make our own replacement to this functionality.

We are using special format of naming of groups defined on mesh in Salome. Groups are read in the code to retrieve materials and simple boundary conditions. For example, by naming groups such as: bottomOxide[SiO2], metalPlate[Dirichlet=1.0] we can assign material and Dirichlet boundary conditions. In MeshAPI, we have a code which

[35]http://en.wikipedia.org/wiki/Simple_API_for_XML
[36]http://www.xmlsoft.org/

```
<?xml version="1.0" encoding="UTF-8"?>
<materialDatabase xmlns="materials.xsd">
 <material name="Si" description="(100)[Silicon]">
  <parameter name="dielectricConstant" value="11.8" />
  <parameter name="longitudeMassForElectrons"
   value="0.98" />
  <parameter name="transversalMassForElectrons"
   value="0.19" />
 </material>
 <material name="SiO2" description="Silicon dioxide">
  <parameter name="dielectricConstant" value="3.9" />
 </material>
 <material name="Air" description="Air">
 </material>
</materialDatabase>
```

Fig. 2. Example of a XML material database file

is able to determine such various characteristics in each node based on group numbers. More complex boundary conditions can be specified in special C++ classes.

E. Our Computational Module

Salome Platform allows us to create custom computational modules. Modules are implemented using Salome and MEDMEM API described in [13]. They can be integrated into Salome Platform (with or without graphical user interface), using automatic conversion using hxx2salome tool, such as described in [14], [15]. We use this possibility to create our own module in C++, which provides methods for simulation of the device. The equations are defined in separate classes and are using DOLFIN and FFC. The module is using MeshAPI and can be extended by defining new classes and can cooperate with other Salome modules.

The module contains special routines, which are automatically wrapped and can be used in Salome, including Salome supervisor module [16] to connect with other simulations or define simulation flow and parameters in Salome. Flow of simulation modules can be defined by using an interactive designer or Python scripting. It is possible to run simulation modules or routines on remote servers.

V. OPERATING SYSTEM AND PORTABILITY

Salome Platform 3.2.6 supports only selected Linux distributions on which runs properly. A more recent version 4.1.4 released in December 2008 is compatible only with recent Debian and Mandriva distributions. Installation of DOLFIN is currently complicated, namely in the case one does not use recent versions of either Debian or Ubuntu. Although there are provided packages for Debian, they cannot be used if one desire optimal performance using PETSc or needs to solve eigenvalues by using SLEPc[37], as in our case.

Salome 3.2.6 does not support and does not compile or run properly on Ubuntu or recent versions of Debian, however it supports Debian Sarge (Debian 3.1), and there are provided pre-compiled binaries of Salome for Debian Sarge. These were the reasons that made us to prefer Debian Sarge as the base for running NanoFEM platform, with Salome Platform and FEniCS/DOLFIN components.

[37]http://acts.nersc.gov/slepc/index.html

Because of our dependency on the Debian Sarge, we have decided to develop and also make possible releases of NanoFEM platform as VmWare[38] virtual machines[39]. The first advantage is that users of the software would not have to perform in general difficult installation of FEniCS/DOLFIN and Salome Platform, which varies on different Linux distributions and on some distributions it is not possible at all. The second advantage is that this way the NanoFEM platform can run also on other operating systems like Windows and Mac OS X, regardless of portability limitations of various used components.

VI. EXAMPLE SIMULATION AND VISUALIZATION OF A FINFET TRANSISTOR

We present an example of simulation in NanoFEM platform. On a 3D structure consisting of 14 geometry groups (allowing to assign different materials and boundary conditions) forming a FinFET transistor, we calculate the Poisson's equation (1) with Dirichlet boundary conditions applied. We had run the simulations on meshes with number of elements in orders of hundreds of thousands of finite elements. Fig. 3 depicts construction of transistor geometry in Salome Platform, Fig. 4 depicts mesh of the transistor consisting with 268.920 finite elements (tetrahedrons) and 46.479 nodes. The simulation on this mesh using the Krylov solver (with biconjugate gradient squared method and with incomplete Cholesky factorization preconditioner), including reading of the mesh, all initializations, preparations of solver and storing of all results takes about 170 seconds (on Intel Core 2 Duo, 1.66 GHz and VmWare on Windows Vista). The simulation process allocates about 220MB of memory. After finishing the simulation, we can use various visualization and post-processing capabilities. For instance, we can visualize scalar map plots of electric potential in the transistor, as is depicted on Fig. 5 and make 3D cut plots, as is on Fig. 6.

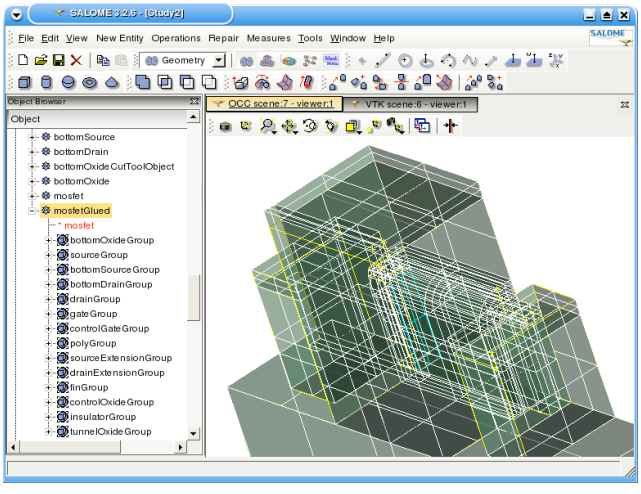

Fig. 3. Modelling geometry of the FinFET transistor

[38]http://www.vmware.com
[39]http://en.wikipedia.org/wiki/Virtual_machine

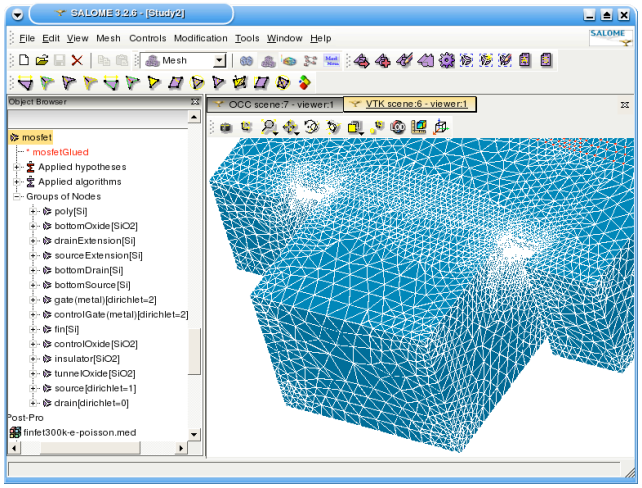

Fig. 4. Automatically generated mesh of the FinFET transistor

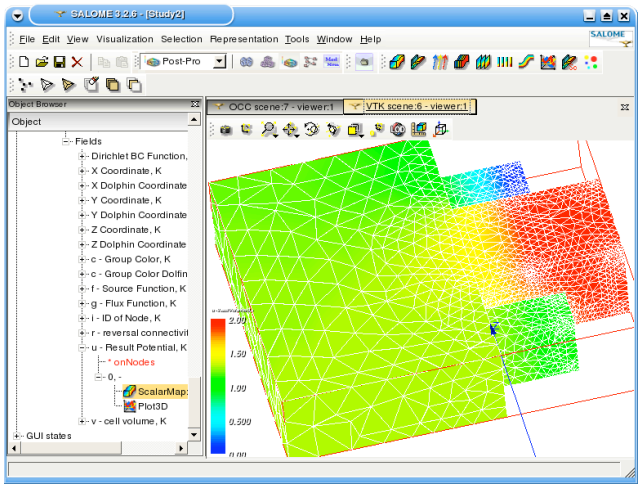

Fig. 5. Scalar map of the electric potential profile in the FinFET transistor

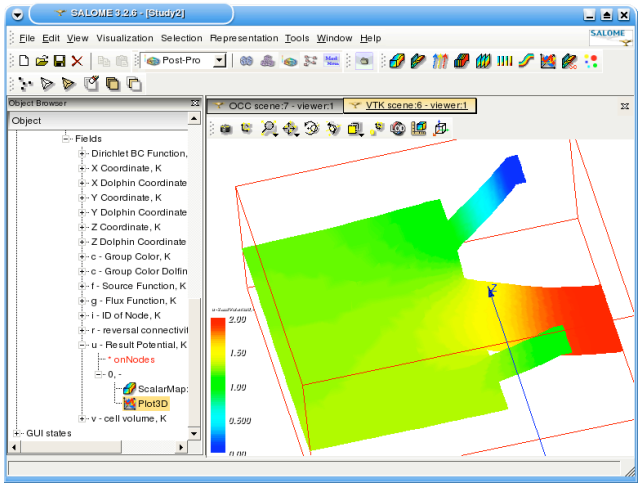

Fig. 6. 3D cut of the electric potential profile in the FinFET transistor

VII. CONCLUSIONS AND FUTURE WORKS

A. Conclusions

NanoFEM platform is a new research environment for TCAD simulations of nanoscale devices based on the finite element method. One of important advantages of the NanoFEM platform proposal is that its two major components - Salome Platform and FEniCS/DOLFIN - are pieces of sophisticated and very high quality open source finite element software, which are under active and long-term external development. Therefore, we do not have to care about very time demanding design, development and maintenance of a finite element pre-processor, post-processor and a finite element simulation library. We can concentrate only on developing and extending of our MeshAPI and computational modules. This part is relatively small and simple and therefore for instance one computer engineer is enough to keep maintenance, extending and coding of the NanoFEM platform (or similar solutions), while physicists can concentrate on mathematical analysis of equations to test on nano-devices and providing them in a variational form format. This allow software developers/engineers and users (e.g. physicists, mathematicians, researchers) of NanoFEM platform to work separately and independently of each other and without necessary expertise in both fields.

The NanoFEM platform design efforts to provide solutions addressing the following goals and requirements:

- Flexibility - simulation user and developers should be able to modify and create physical parameters, boundary conditions or even whole governing equations;
- Simple, yet powerful definition of the solved problem allowing to concentrate on defining equations to solve, rather than necessity to manually program finite elements;
- Interactive pre-processing and post-processing (namely geometry definition, mesh generation, visualization);
- Automatic generation of meshes suitable for FEM;
- Ability to control flow of simulation modules (either with an interactive designer or a scripting language);
- Ability to run simulation modules and methods on remote servers;
- High performance due to usage of performance optimal, quality open source components and libraries;
- Control of sparse linear algebra solver methods;
- Usage of standard formats for exchanging and storing simulation cases and results;
- Support for XML material database;
- Good extendibility and modularity;
- Portability – using virtualization possibilities to be able to use the software on different operating systems, regardless of eventual portability limitations of various used code components.

We believe that others could eventually consider to build similar environments as our NanoFEM platform for general finite element simulations. The core necessary components, Salome Platform and FEniCS/DOLFIN are available under the free LGPL license. Although we currently do not provide MeshAPI source codes and do not have a public release of NanoFEM platform, others could still implement their own similar codes aimed to connect Salome Platform and FEniCS/DOLFIN and compute their own scientific equations and create similar solutions as ours.

B. Future Works

There are many options for further development of NanoFEM platform. We would like to test and implement various complex equations, test complex simulation flow and coupling scenarios, test advanced exchanging of data between modules using Salome Platform supervision module, design and implement complex boundary conditions specification, either by using XML files, or by using Salome data module for these features, when they will be available. We would like to make native Debian and also Windows versions (when all used components will be ready for such eventual port), without necessity to use virtualization.

REFERENCES

[1] M. A. der Maur, M. Povolotskyi, F. Sacconi, G. Romano, E. Petrolati, and A. D. Carlo, "Multiscale simulation of electronic and opto-electronic devices with TiberCAD," in *Simulation of Semiconductor Processes and Devices* (T. Grasser and S. Selberherr, eds.), pp. 245–248, Springer Vienna, 2007.

[2] J. Schöberl, "NETGEN: An advancing front 2D/3D-mesh generator based on abstract rules," *Computing and Visualization in Science*, vol. 1, no. 1, pp. 41–52, 1997.

[3] H. Si, *A Quality Tetrahedral Mesh Generator and Three-Dimensional Delaunay Triangulator*. http://tetgen.berlios.de, 2006.

[4] C. Geuzaine and J.-F. cois Remacle, "Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities," *International Journal for Numerical Methods in Engineering*, 2009.

[5] J. Treibig, S. Berler, and U. Rüde, "ORCAN: A platform for complex parallel simulation software," in *ARCS Workshops* (W. Karl, J. Becker, K.-E. Großpietsch, C. Hochberger, and E. Maehle, eds.), vol. 81 of *LNI*, pp. 295–304, GI, 2006.

[6] T. Dupont, J. Hoffman, C. Johnson, R. C. Kirby, M. G. Larson, A. Logg, and L. R. Scott, "The FEniCS project," Tech. Rep. 2003–21, Chalmers Finite Element Center Preprint Series, 2003.

[7] B. S. Kirk, J. W. Peterson, R. H. Stogner, and G. F. Carey, "libMesh: a C++ library for parallel adaptive mesh refinement/coarsening simulations," *Engineering with Computers*, vol. 22, no. 3, pp. 237–254, 2006.

[8] A. Logg, "Automating the finite element method," *Archives of Computational Methods in Engineering*, vol. 14, no. 2, pp. 93–138, 2007.

[9] R. C. Kirby and A. Logg, "A compiler for variational forms," *ACM Transactions on Mathematical Software*, vol. 32, no. 3, pp. 417–444, 2006.

[10] M. S. Alnaes, H.-P. Langtangen, A.Logg, K.-A. Mardal, and O. Skavhaug, *UFC Specification and User Manual*, 2008. http://www.fenics.org/ufc/.

[11] A. Logg, G. N. Wells, *et al.*, *DOLFIN User Manual*, 2009. http://www.fenics.org/dolfin/.

[12] A. Logg, *FFC User Manual*, 2007. http://www.fenics.org/ffc/.

[13] V. Bergeaud, N. Bouhamou, N. Crouzet, E. Fayolle, P. Goldbronn, and J. Roy, *MEDMEM user's guide*, 2008. http://www.salome-platform.org.

[14] V. Bergeaud, *SALOME component integration tutorial*, 2008. http://www.salome-platform.org.

[15] *hxx2salome : a Salome component generator*. http://www.salome-platform.org/, 2008.

[16] A. Ribes and C. Caremoli, "Salome platform component model for numerical simulation," in *COMPSAC '07: Proceedings of the 31st Annual International Computer Software and Applications Conference*, (Washington, DC, USA), pp. 553–564, IEEE Computer Society, 2007.