MDS – A New, Highly Extensible Device Simulator

T. D. Linton, Jr., K. Foley, F. Heinz, R. Kotlyar, P. Matagne, A. Eremenko, S. Sergienko, M. Stettler, M. D. Giles, and B. Voinov

Intel Corporation, 2200 Mission College Blvd., Santa Clara, CA
tom.linton@intel.com

Abstract

Device simulation needs are growing more diverse and it is difficult for traditional simulators to satisfy them while maintaining usability, maintainability, speed, and robustness. The Modular Device Simulator (MDS) is a completely new simulator framework that addresses this problem by providing simulation building blocks within a dynamic, runtime-configurable framework driven by a scriptable input parser. This flexible framework allows MDS to be applied to a wide range of problems that traditionally would have been handled by many independent codes. MDS has been applied to the 45 nm node and beyond, including advanced applications such as Schrödinger/drift-diffusion and non-equilibrium Green's function (NEGF).

1 MDS Architecture

MDS is written in C++ and is based on a set of standard objects: Solvers, Domains and Fields. A Solver interacts with a set of Domains or subsolvers through a strict protocol to assemble and solve a particular type of problem. A Poisson simulation, for example, might use a Newton Solver that operates on a set of electrode, interface, Laplace, and Poisson domains. The Newton Solver calls on each of these Domains as it assembles a global Jacobian and then solves a series of bias points under control of a boundary condition iterator object. New Solvers can be added that either use existing Domains or that use new kinds of Domains, possibly with new Domain/Solver protocols.

A Domain assembles part of a problem for a Solver. In MDS, each material region has a separate mesh, and one or more Domains are typically defined for each electrode, interface, or bulk mesh. Domains can also be defined across mesh regions for non-local phenomena such as tunneling.

A Field represents a mesh-based quantity such as a device model or solution variable for a single Domain. Different element types (integer, real, vector, etc.) are possible, as are different Field value locations (node-based, edge-based, element-based, etc.). A Field can have arguments, i.e., other Fields that it uses in computing itself. The primary function of a Field is "update", which takes an argument specifying what to update, including the field value itself and possibly derivatives. This function recomputes just what is needed, based on timestamps of itself and its arguments. MDS dynamically creates and hooks up just those Fields that are actually needed by instantiated Domains.

2 Configuration And Operation

A key feature of MDS is its dynamic configuration, which is largely enabled by the parameter database (pdb), a tree structure in memory organized somewhat like the Unix filesystem, for getting and setting values. Example value types include integer, real, string, and vector. Pdb creation areas define default configuration information for MDS objects, and each instantiated MDS primary object is also provided a runtime pdb area where it can read and write values for communication with other objects. The pdb is initialized from a file and modified during runtime to further customize system configuration.

MDS uses a Tcl input parser. Commands provide a high-level interface similar in appearance to legacy simulators, but are also able to directly access Fields, the pdb, and other data that is inaccessible in traditional simulators.

The MdsUserLib is an incr Tcl library that automates set-up and execution of simulations. Its key object is the MdsSim, which instantiates Domains and Solvers based on rules in the pdb. The MdsSim is subclassed for different kinds of simulations. Examples include the PoissonSim for Poisson-only and DD2CarrierSim for two-carrier device simulations.

3 MDS Applications

The extensibility and utility of the MDS system is clearly demonstrated in a self-consistent Schrödinger drift-diffusion simulation of a Tri-Gate transistor[1]. Here a particular simulator object, the PCCSSim, instantiates 3D bulk Poisson, 3D bulk Laplace, 3D bulk current-continuity, interface, and electrode Domains under control of a Newton Solver, as in a standard 3D drift-diffusion simulation. It also instantiates a set of 2D Schrödinger Domains and Solvers on slice meshes at cross-sections along the length of the device. The scripted PCCSSim object implements a predictor-corrector algorithm that successively calls the Poisson/drift-diffusion and Schrödinger Solvers. During iteration, slice Fields present 2D views of the potential to the Schrödinger Domains and a composite Field assembles quantum correction values on each 2D slice into a single 3D Field for the Poisson and current continuity Domains, as shown in Figure 1. Corrections are calculated with a 3-valley conduction band model for the electrons.

Much of the power of MDS comes from its ability to implement a totally new type of simulator such as the PCCSSim predictor-corrector by dynamically interconnecting standard Domains, Fields, and Solvers at runtime using new incr Tcl objects. Solution control and data communication between objects make use of standard protocols and mechanisms from the MDS system. In addition, the high-level algorithm is written in an easily modified scripting language without significant performance penalty because actual computation takes place in the compiled Domain, Field, and Solver objects.

Figure 2 shows the Tri-Gate device with conduction band energies at bias and Figure 3 shows resulting Schrödinger densities. Other types of simulations are also available, such as nonisothermal, hydrodynamic, K•P, and NEGF. Many structures have been simulated, including MOS, flash, nanowire, and nanotube.

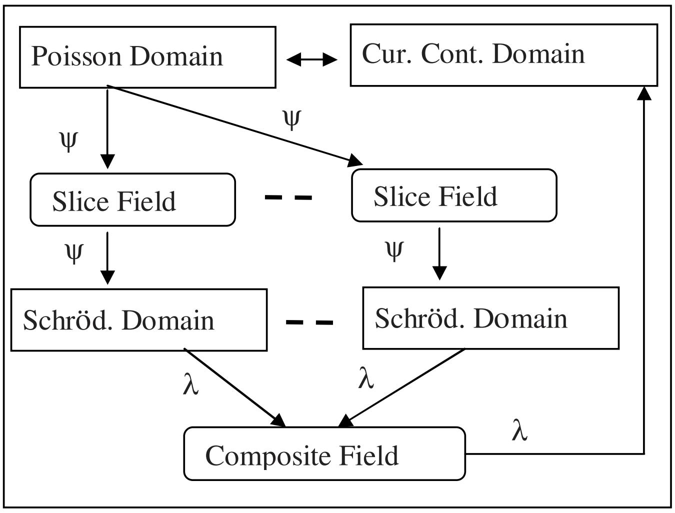

Figure 1: Slice Fields pass potential ψ to Schrödinger Domains; Composite Field assembles correction factor λ for drift-diffusion Domains.

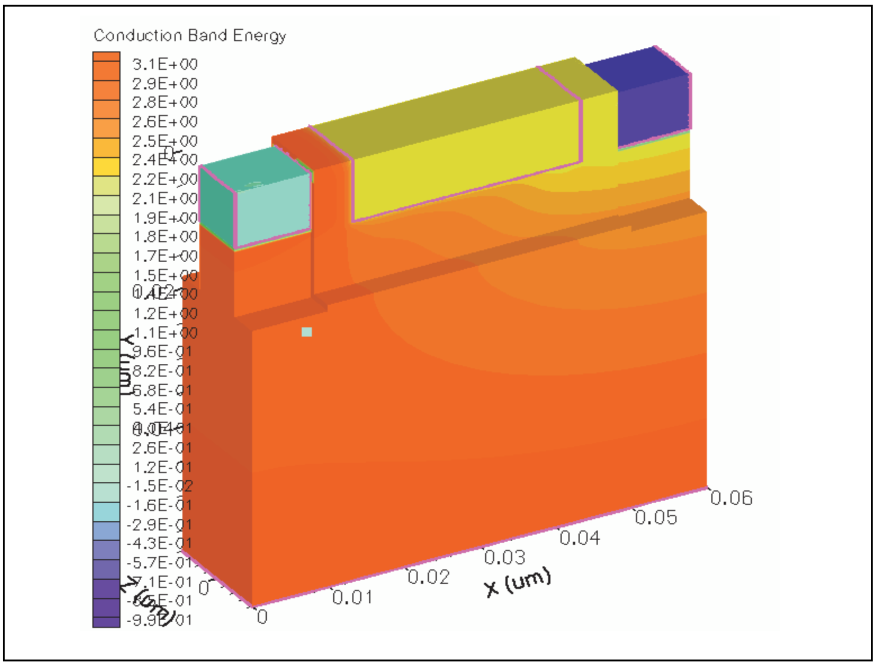

Figure 2: Tri-Gate structure, showing conduction band energy.

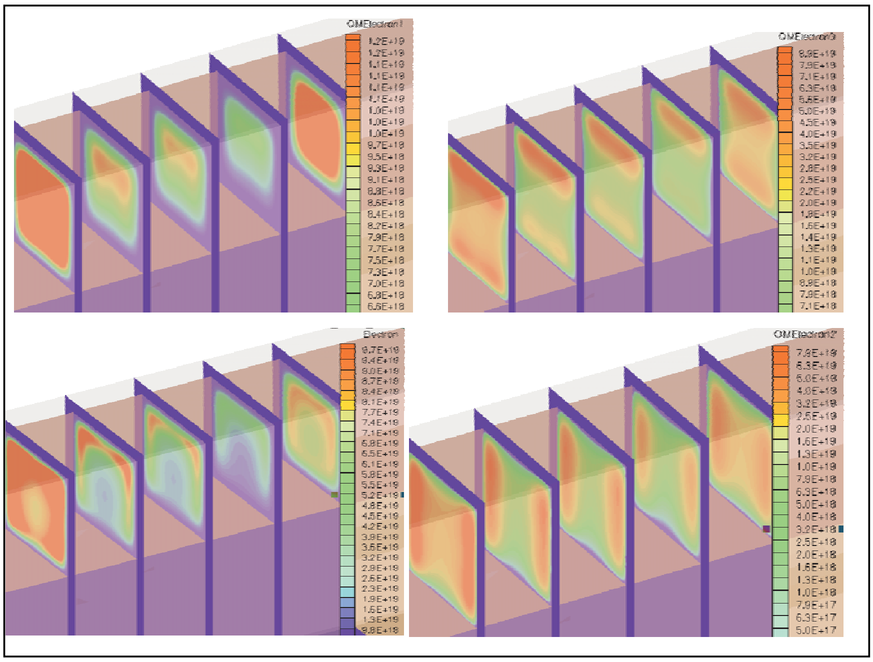

Figure 3: Schrödinger densities are shown clockwise from upper left for <100>, <001>, and <010> valleys; and total.

4 Conclusion

MDS has introduced a new architecture with proven extensibility for advanced studies. It has demonstrated robustness and efficiency in production use for the 45 nm node and provides the basis for a wide range of future applications.

References

[1] J. Kavalieros, B. Doyle, S. Datta, G. Dewey, M. Doczy, B. Jin, D. Lionberger, M. Metz, W. Rachmady, M. Radosavljevic, U. Shah, N. Zelic, R. Chau, *Tri-Gate Transistor Architecture with High-k Gate Dielectrics, Metal Gates and Strain Engineering*, Digest of Technical Papers, 2006 Symposium on VLSI Technology (2006).