Dynamic mesh for TCAD modeling with ECORCE

A Michez¹, J Boch¹, A Touboul¹, F Saigné¹
¹Université de Montpellier, IES – UMR UM/CNRS 5214, 860 Rue Saint Priest, 34095 Montpellier Cedex 5, France
E-mail: alain.michez@umontpellier.fr

Abstract. Mesh generation for TCAD modeling is challenging. Because densities of carriers can change by several orders of magnitude in thin areas, a significant change of the solution can be observed for two very similar meshes. The mesh must be defined at best to minimize this change. To address this issue, a criterion based on polynomial interpolation on adjacent nodes is proposed that adjusts accurately the mesh to the gradients of Degrees of Freedom. Furthermore, a dynamic mesh that follows changes of DF in DC and transient mode is a powerful tool for TCAD users. But, in transient modeling, adding nodes to a mesh induces oscillations in the solution that appears as spikes at the current collected at the contacts. This paper proposes two schemes that solve this problem. Examples show that using these techniques, the dynamic mesh generator of the TCAD tool ECORCE handle semiconductors devices in DC and transient mode.

1. Introduction
TCAD (Technology Computer Aided Design) modeling is widely used to design semiconductor devices and simulate their behaviour. The use of TCAD software is uneasy and most often at least one engineer of a team is dedicated to this work. The mesh design is the main problem since, to compute a true solution, the mesh must accurately follow the gradients of the 3 Degrees of Freedom (DF): potential and densities for electrons and holes. Indeed, carrier densities in semiconductors can change by several orders of magnitude inside depletion regions. These regions can be very thin depending on the doping level and need a highly refined mesh. Conversely, areas where doping are constant represent the main surface of the component and will only need a coarse mesh. Furthermore external events like varying bias or ionizing radiations induce fast and substantial changes in the spatial distribution of carriers.

However, for now, meshes in TCAD software are manually defined and can not be modified during modeling. Then the designer of a mesh must take into account all distributions of DF that will occur during modeling to create the fixed mesh. For this he adds nodes that will be unnecessary for most steps of the simulation and that will increase CPU time. Anyway, the task is so complex that a human operator can at best get an approximate result.

Tools that automatically create and modify a mesh according to gradients of DF during modeling are already used in others fields of physics like fluid and solid mechanics [1][2]. These dynamic mesh generators insure users to get the best balance between precision of solution and CPU time.

But, as explained previously, a dynamic mesh generator for TCAD modeling will face harsh numerical constraints. To correctly define gradients of DF that change by several orders of magnitude in thin areas, we need a reliable criterion that predicts when adding or when removing a node. Adding
too late or removing too early a node triggers oscillations in the solution that result in numerical noise at the electrode currents. Furthermore DF of added nodes must be carefully calculated especially for transient modeling where time derivative of DF depends on previous steps.

We showed in a previous paper that ECORCE [3] is the first TCAD software that provides a dynamic mesh for transient analysis [4]. In this paper we describe the criterion used by ECORCE to drive the refinement and coarsening of the mesh and the calculation strategy for added nodes.

2. Overview of ECORCE
ECORCE is a TCAD tool that provides a Graphical User Interface (GUI) and a solver, including a dynamic mesh generator, integrated in one application. The whole software is developed in C++. The GUI is based on Qt [5] and offers a friendly interface for all steps of modeling: designing of model, execution of calculation, display and analyse of results. The main part of ECORCE is distributed under the GNU General Public License (GPL) [3].

ECORCE uses a well-known drift-diffusion model [4,6] coupled with heat equation. It generates structured meshes build with rectangular elements and groups of three triangular elements for connecting rectangular elements of different sizes as shown in Figure 1. Our meshing algorithm prevents the creation of obtuse triangles that are known to generate inaccurate results and numerical instabilities and ensures that all angles will be less than or equal to 90°.

To ease changes of the mesh, data are structured using 3 objects: .node, edge and face. Each object can access adjacent objects via a list of pointers as shown in Figure 2.

![Figure 1. Example of mesh generated by ECORCE](image)

![Figure 2. Connections between objects of the mesh](image)

3. Criterion to refine and coarsen the mesh
DF are assumed to vary linearly within an element when discretizing differential equations. This approximation is correct as long as the gradient is small enough between two consecutive elements.

![Figure 3. Example of adjacent edges and values of DF associated to each node.](image)

![Figure 4. Quadratic interpolation on nodes 1, 2 3 (red dotted line) and on node 2, 3, 4 (green dashed line). Differences between linear and quadratic interpolations are computer at point $x_m=(x_3+x_2)/2$ (arrow).](image)
Then, to correct the mesh, we need an estimate of the error induced by the linear interpolation on one edge. This estimation can be done by calculating the value in the middle of the edge using a quadratic interpolation and comparing with the same value calculated with a linear interpolation. For this we need to retrieve adjacent edges (Figure 3). Thank to our structured data, this can be easily done including two dimension modeling.

Interpolating value in the middle of the edge 32 assuming a linear interpolation between nodes 2 and 3 gives:

\[ V_{32} = \frac{V_3 + V_2}{2} \]  

(1)

Interpolating the same value assuming a quadratic interpolation between nodes 1, 2, 3 (Figure 4: red dotted line) and between nodes 2, 3, 4 (Figure 4: green dashed line) gives:

\[ V_{321} = L_1(x_m) V_1 + L_2(x_m) V_2 + L_3(x_m) V_3 \]

with

\[ L_i(x) = \prod_{j=1, j \neq i}^{3} \frac{x-x_j}{x_i-x_j} \]  

(2)

\[ V_{432} = N_2(x_m) V_2 + N_3(x_m) V_3 + N_4(x_m) V_4 \]

with

\[ N_i(x) = \prod_{j=2, j \neq i}^{4} \frac{x-x_j}{x_i-x_j} \]  

(3)

\[ L_1(x), L_2(x), L_3(x) \] are the Lagrange polynomials of order 2 on nodes 1, 2, 3 and \[ N_1(x), N_2(x), N_3(x) \] are the Lagrange polynomials of order 2 on nodes 2, 3, 4.

Then the relative error of one DF on the edge 32 can be expressed by the differences:

\[ E_{321}^{DF} = 2 \frac{|V_{321} - V_{32}|}{V_{321} + V_{32}} \]

and

\[ E_{432}^{DF} = 2 \frac{|V_{432} - V_{32}|}{V_{432} + V_{32}} \]  

(4)

and we keep the higher of these 2 errors:

\[ E_{32}^{DF} = \max(E_{321}^{DF}, E_{432}^{DF}) \]  

(5)

For the DF whose value changes slightly over the device, like potential (psi) and temperature (T), this formula can be used as is. But for the DF that can change by several orders of magnitude like electron and hole densities, the relative error is computed using the decimal logarithm of the DF. Then the formula is weighted to reduce the refinement generated by low values that have a negligible impact on the global solution. The weight function (hereinafter called W) is a spline function as shown on Figure 5.

![Figure 5. Weight function to reduce refinement for low values of electron and hole densities.](image)

Three parameters of the weight function can be adjusted: the precision threshold (P_{th}) for n or p densities, the slope at the precision threshold point (S_{th}) and the slope at the maximum doping value (S_{max}). These values have been adjusted to optimize the refinement ratio between high carrier densities (at maximum doping value) and low carrier densities (which can be considered negligible):
\( P_{th} = 10^8 \, \text{cm}^{-3}, \, S_{th} = 0 \) and \( S_{\text{max}} = 0.5 \)

With the weight function, the relative error formulas for \( n \) and \( p \) densities are:

\[
E_{32}^n = \max(E_{321}^n, E_{432}^n)W(n) \quad E_{32}^p = \max(E_{321}^p, E_{432}^p)W(p)
\]

and we keep the higher errors on all DF:

\[
E_{32} = \max(E_{321}^{\text{ref}, E_{321}^{Fr}}, E_{32}^p, E_{322}^p)
\]

Thus, the dynamic mesh generation in ECORCE is driven by four parameters:

- the minimum size of edges (\( \text{Size}_{\text{min}} \))
- the maximum size of edges (\( \text{Size}_{\text{max}} \))
- the relative precision for edge splitting (\( P_{\text{refine}} \))
- the relative precision for edge merging (\( P_{\text{release}} \))

With these four parameters we define the conditions to refine one edge or merge two edges. An edge is refined if:

\[
\left( \text{Length} > \text{Size}_{\text{max}} \right) \text{OR} \left( \left( E_{\text{edge}} > P_{\text{refine}} \right) \text{AND} \left( \text{Length} > 2 \times \text{Size}_{\text{min}} \right) \right)
\]

where \( \text{Length} \) and \( E_{\text{edge}} \) define the edge length and the higher relative error for this edge (on all DF).

Two edges (numbered 1 and 2) are merged if:

\[
\left( E_{\text{edge1}} < P_{\text{refine}} \times P_{\text{release}} \right) \text{AND} \left( E_{\text{edge2}} < P_{\text{refine}} \times P_{\text{release}} \right) \text{AND} \left( \text{Length1} + \text{Length2} < \text{Size}_{\text{max}} \right)
\]

where \( \text{Length1}, \text{Length2}, E_{\text{edge1}} \) and \( E_{\text{edge2}} \) define edge 1 length; edge 2 length, the higher relative error of edge 1 and the higher relative error of edge 2.

Consistent values for the relative precisions are 10% to 0.1% for \( P_{\text{refine}} \) and 10% to 0.1% for \( P_{\text{release}} \). Low values of \( P_{\text{refine}} \) result in coarse meshes and low CPU time. High values of \( P_{\text{refine}} \) give finer meshes but higher CPU time.

Too high values of \( P_{\text{release}} \) (more than 10%) can lead to oscillations between mesh refinement and release, increasing the CPU time for the mesh calculation. Too low values of \( P_{\text{release}} \) limit the removal of nodes, increasing the CPU time.

4. Calculation strategy for DF of new nodes

Adding one node, even in a device for which the mesh is properly designed, can bring a significant change on the entire solution. As an example consider the PN junction presented Figure 6 and the electron and hole densities in the junction for a -1V reverse bias presented Figure 7.

![Figure 6. Example of a PN junction](image)

For this modeling coarsening of the mesh is disabled and others parameters are: \( \text{Size}_{\text{min}} = 10^{-3} \, \mu m \), \( P_{\text{refine}} = 10^{-2} \) and \( \text{Size}_{\text{max}} = 10 \, \mu m \), resulting in a 95 nodes mesh. Then performing the same modeling, adding only one node at the position \( x = 4.96 \, \mu m \) (as shown in Figure 7), we get the change for electron and hole densities displayed on Figure 8. Since the node have been added inside the space charge region we can expect to see a change only in this region, namely for low electron and hole densities.

But we see on Figure 8 that this change is also significant for high electron and hole densities. This is highlighted on Figure 9 which presents the total electrical charge density and the difference of...
electrical charge density between the two modeling. The node is added at the position 4.96µm and the electric charge shows a significant change up to 2µm from that point. Thus adding one node in a device affects nodes over a wide area around the insertion point. This is not a problem for DC modeling where solution can be recomputed after each change of the mesh. But for transient modeling, derivative as a function of time of electron and hole densities are calculated from previous time steps. As shown on Figure 10 n steps are needed to calculate time derivatives at time \( t_0 \). \( n \) depends on the order of the interpolation polynomial [7]. Then DF of the \( n \) steps for each node of the affected area (Figure 9) become unsettled. If the solution is not corrected, the next steps of the modeling may be flawed.

**Figure 7.** Electron and hole densities in the PN junction

**Figure 8.** Change of electron and hole densities after adding one node at the position \( x=4.96\mu m \)

**Figure 9.** Charge density in the PN junction and change of this density when adding one node at position \( x=4.96\mu m \)

**Figure 10.** Time steps used to compute \( n \) and \( p \) densities derivative as a function of time.
4.1. First scheme
As a first method, after a mesh change in transient mode, DF of new nodes for steps 1 to n-1 are linearly interpolated using the DF of the split edge. Then, the complete solution for step 0 is recalculated, namely all DF for all nodes are recomputed for this step. With this method, if the mesh is not fine enough, adding nodes on steps 1 to n-1 brings a too high change in the solution in the affected area. Then the derivatives of electron and hole densities as a function of time show oscillations that appear as spikes in the current collected at the contact. Figure 11a displays the current crossing the PN junction for a sinusoidal bias applied on the Anode with a frequency of 20MHz, amplitude of 0.6V and a 3rd order polynomial interpolation for time derivative. At this frequency, the diode behaves roughly as a capacitor. Thus, the shape of the current is correct but spikes induced by added nodes are clear.

Applying tighter parameters to refine the mesh solves the problem through an increase of the node number and thus an increase of the CPU time. Spikes disappear on Figure 11b but we need 3 to 4 times more nodes than for results displayed Figure 11a.

4.2. Second scheme
Another way to solve this problem is to linearly interpolate DF of new nodes only for step n-1. Then the solution of step n-2 can be recalculated using step n-1 to compute the time derivative $\frac{dn}{dt}$ and $\frac{dp}{dt}$ by a first order interpolation. Step n-3 can then be recalculated using a 2nd order interpolation for time derivative using step n-1 and n-2. And so by degrees the full chain of steps needed to compute step 0 is recalculated.

With this scheme, for each mesh change, the DF for all nodes from step n-2 to 0 are recomputed, especially in the affected area. This method increases the CPU time, but much less nodes are needed to get an accurate solution. Figure 12a displays the current and number of node for this scheme for the same bias and mesh parameters than those of Figure 11a. On Figure 12a the spikes are hardly visible for the same number of nodes than Figure 11a. Furthermore by tightening slightly the mesh parameters (Figure 12b) the current shows no more defects for 2 to 3 times less nodes than for Figure 11b.

**Figure 11.** Current density and number of nodes for a PN junction with a sinusoidal bias on the Anode, frequency: 20 MHz, amplitude: 0.6V, for the first scheme.
5. Conclusions

Meshing is a critical step of TCAD modeling. From the mesh quality depend the solution precision and the CPU time. But meshing in TCAD software is for now hand made and only one grid is used for all the steps of the simulation. Yet spatial distribution of DF change during modeling and the mesh should be adjusted to gradient of DF at each step. A dynamic mesh generator is a good solution to solve this problem. But driving the mesh generator needs a criterion that tells when to add nodes and when to remove nodes. Thank to the structured data of the mesh, ECORCE can easily retrieve adjacent nodes. Thus a criterion based on polynomial interpolation between 4 adjacent nodes gives a good idea of the DF gradients and allows a fine adjustment of the mesh.

Furthermore, use of dynamic mesh in transient mode with TCAD modeling is challenging. Because electron and hole densities can change by several orders of magnitude in depletion zone, adding only one node in such a place changes the solution in a large area around the insertion point. Then in transient modeling, changing the mesh induces oscillations in the solution that appears as spikes on the current collected at the contacts. Two schemes are proposed to solve this problem. One requires more nodes but less solving cycles, while the other one needs less nodes but more solving cycles.

Using the dynamic mesh generator, the criterion to adjust the mesh and the schemes to remove oscillations of the solution in transient modeling, ECORCE suppresses the hardest step of TCAD modeling.

References

[1] Hudson B 2007 Dynamic mesh refinement (Carnegie Mellon University)
[2] Lopez D, Bruno L , Dynamic Mesh Optimization for Free Surfaces in Fluid Simulation, presented at IMR - 21th international meshing roundtable - 2012, p. 1–5.
[3] ECORCE Website (http://www.ecorce.eu/)
[4] Michez A, Dhombres S and Boch J 2015 ECORCE: A TCAD Tool for Total Ionizing Dose and Single Event Effect Modeling Nucl. Sci. IEEE Trans. On 62 1516–27
[5] Qt Website (http://www.qt.io/)
[6] Sze S M and Ng K K 2006 Physics of semiconductor devices (John Wiley & Sons)
[7] Michez A and Bordure G 1991 Transient simulation of silicon devices under high carrier injection. Comparison of various time stepping schemes. COMPEL Int. J. Comput. Math. Electr. 10 231–40