Background

A power IGBT (Insulated Gate Bipolar Transistor) is conventionally made up of a repetitive array of homogenous IGBT cells. Such a homogenous configuration renders a uniform current flow across the active surface area of the IGBT chip when the IGBT is turned on. Under a short-circuit condition, however, the IGBT being turned on is exposed to a very high collector-to-emitter voltage. In this condition, the IGBT conducts a very high collector current, leading to correspondingly high power dissipation in the form of heat flowing uniformly across the chip. If the heating of the chip exceeds a critical level during a short-circuit operation, the device may fail or even get destroyed by local overheating in conjunction with the establishment of current filaments in a localized area, or the hot spots, within the device.

Modeling and Simulation

In cell mode of Victory Process an IGBT array consisting of 8 cells, as shown in Figure 1, is generated by joining 8 copies of the IGBT single cell together. This IGBT multicell array is uniform along its width of 40 µm. Each IGBT single cell is 20 µm long and features a 1.3 kV trench-gate design with a fieldstop layer (n-buffer) that consumes 16 µm out of the n-drift region thickness of 134 µm. For the sake of convenience, the doping profiles of the cell structure are modeled with analytic functions.

For simulation of filament formation, the multicell IGBT is tested with the test circuit pictured in Figure 2. The collector-to-emitter terminals are supplied with a constant voltage source of 600 V while the gate is pulsed from 0 V to 15 V in 10 ns with a pulse length of 10 us.

Figure 1. Simulated 8-cell field-stop trench-gate IGBT array with a width of 40 µm.

Continued on page 2...
During the short-circuit test the IGBT is assumed to be mounted on a heat sink maintained at an ambient temperature of 300 K. This can be modeled in Victory Device by adding a thermal contact with a thermal resistance of 0.3 cm²K/W to the heat sink to the bottom collector electrode.

To predict the short-circuit behavior in the multicell array, Victory Device takes the following physical models into account: (1) low-field mobility dependent on doping and perpendicular electric field, (2) parallel-electric-field dependent mobility (velocity saturation), (3) Shockley-Read-Hall and Auger recombination, (4) impact ionization and (5) self-heating effects.

Simulation Results

The simulated short-circuit waveforms of the 8-cell IGBT in Figure 3(a) indicate that the device can withstand the power pumped into it for 5.57 µs before it fails. Furthermore, a peak in the current waveform at t = 0.48 µs and a kink at t = 4.5 µs suggest two possible occasions of nondestructive filament formation prior to device burn-out. Figure 3(b) captures the 3D electron current density...
distribution at the failure point in time \( t = 5.57 \mu s \). At that time, the hot spot current filaments are concentrated mainly on the emitter side of the eighth cell to the right of the multicell array and the current density is non-uniformly distributed across the device from the hot spots.

To illustrate how current filaments evolve shortly before excessive heat burns out the device, the same 8-cell IGBT with a reduced width of 1 µm is simulated. Such a 1 µm wide multicell array can be thought of as a quasi-3D multicell array. Similar behavior can be observed in the short-circuit waveforms of the quasi-3D 8-cell IGBT shown in Figure 4(a). The quasi-3D multicell array fails at a short-circuit turn-on time of 5.24 µs. Figure 4(b) displays cross-sectional views of current filament evolution in the quasi-3D 8-cell IGBT for a number of selected points in time prior to and upon device failure at \( t = 5.24 \mu s \). It is evident that, in the vicinity of the failure point, the filament formation is so intense that the current crowding occurs at the right edge of the multicell array. The current filaments are gradually widened and, simultaneously, establish a front that moves towards the left edge of the multicell array, starting from the collector side, until the device is burnt out.

Summary:

With the combined advanced features of Victory Process and Victory Device including:

- 3D rapid prototyping
- Electrothermal computation in 3D
- Mixed-mode 3D device and circuit simulation
- Fast and robust 3D direct solver using MPI

Silvaco’s 3D TCAD tools make it possible for large complex power semiconductor structures to be efficiently handled. More importantly, the 3D device simulator demonstrates for the first time that current filamentation phenomena in a 3D IGBT multicell array subject to a short-circuit condition can be simulated in days and not weeks or months. This clearly facilitates the use of 3D TCAD simulation for understanding of physical processes in power devices, as well as optimization and prediction of device characteristics.

References:

Optical Simulation of Organic Light Emitting Diode by Transfer Matrix Method with a Green’s Function Approach and 2D FDTD

Introduction

Organic light emitting diode (OLED) has been getting much attention over the past decades in the field of displays and lighting applications for its excellent efficiency, color quality and color tunability. Optical modeling of OLEDs is one important development issue to create high performance devices [1]. In this article, several optical simulations of OLEDs are presented by using recently implemented features in Atlas. First, we present the multiple dipole sources emission from a stacked layers structure, including the interference effect between the emitted light and reflected light using Transfer Matrix Method (TMM) with a Green’s Function Approach [2-4]. Then, we present the 2D FDTD analysis on the device with the grating structure using the finite difference time domain (FDTD) [5].

1. Multiple Dipole Sources Emission

White OLEDs are emerging and expected as the next generation light sources in the field of AM-OLED TVs and lighting devices. Various device structures have been proposed to realize white light emission [1]. We have selected a multiple stacked layers structure with/without mixtures of coherent and incoherent layers, and performed analysis of multiple dipole sources emission using TMM with Green’s function.

1.1 Coherent case

A stacked layers structure for the analysis of the multiple dipole emission shown Figure 1 is taken from the reference paper [6].

The multilayer structure of OLED device composed of 8 coherent layers with 3 emissive dipoles is created using Radiant [7], which is an interactive tool to simulate LED and OLED devices. Three point dipoles are set in the center of each emitting layer (EML_Dipole1, EML_Dipole2 and EML_Dipole3) and the spontaneous emission rate for each dipole is set to 1. Photo Luminescence data corresponding to each layer is selected from the Silvaco Spectrum Library, Alq3.spc, DCM2Alq3.spc and Alq3_PtOEpspc are set to EML_Dipole1, EML_Dipole2 and EML_Dipole3, respectively.

In the syntax of Atlas, these settings are defined by the “MATERIAL” statement as follows.

```plaintext
material region=EML_Dipole3 emit.num=1 emit.rate=1 emit.file=Alq3_PtOEpspc out.uspec=PL_red.spc
```

The TMM Green’s Function method is enabled by the “TMM.GF” parameter on the “SAVE” statement, and solves the 3D field of a point dipole inside the stack. The dipole is assumed to be the Hertzian and randomly oriented for the coherent simulation. Output spectral power density at viewing angle 0 degree is calculated.

In the syntax of Atlas, these settings are defined by “SAVE” statement as follows.

```plaintext
save tmm.gf ^emit.bot bot.hs numrays=1 view.min=0 view.max=0 emin=1.5 emax=3.5 nsamp=501 angpower=as0.log out.spect=pspect0.log spect.angle=spect_angle0.log ^calc.emission.fac dipole.polar=45 theta=45 horizontal=0.666
```

![Figure 1. Multilayer structure device with 3 point dipoles inside.](image1)

![Figure 2. Simulated output spectral power densities at viewing angle 0 degree for the multiple dipole sources.](image2)
Simulated output spectral power density for the multiple dipole sources is shown in Figure 2. Total output spectral power density is calculated by summing up the power density emitted by each dipole source. The profile can be controlled and calibrated by changing the spontaneous emission rate of the dipoles, and the contribution of each dipole source can be estimated.

1.2 Incoherent case

The multilayer structure of OLED device for the incoherent case is shown in Figure 3.

A thicker oxide layer of 10 microns is placed on top of the structure shown in Figure 1 as to investigate the effect of the interference by the emitted light and reflected light within the device. Three point dipoles are set in the center of the emitting layer and the emitting rate for each dipole is the same as the coherent case in the previous section. Output spectral power density at viewing angle 0 degree is calculated.

The incoherent mode is enabled by the “INCOHERENT” parameter on the “SAVE” statement. If this parameter is activated, the interference within the specified layers by the “MATERIAL” statement is not considered.

In the syntax of Atlas, these settings are defined by “MATERIAL” and “SAVE” statement as follows.

```
material material=oxide real.index=1.5 imag.
index=0.0  incoherent
save tmm.gf ^emit.bot bot.hs numrays=1 view.
min=0 view.max=0 emin=1.5 emax=3.5 nsamp=501
angpower=as0.log out.spect=pспект0_adj.log spect.
angle=spect_angle0.log ^calc.emission.fac dipole
polar=45 theta=45 horizontal=0.666 normalize.pl
normalize.spec incoherent
```

Simulated output spectral power densities with and without considering the effect of the interference are shown in Figure 4. Fine fringe patterns appeared on the spectrum if the interference within the layers is not considered.

2. 2D FDTD

The typical OLED device has a periodic layout. Recently, the layout size of the OLED or driving TFT is scaled to smaller sizes. If the layout size is closer to the wavelength of the incident light from outside of the device, the interference effect between the incident light and reflected light cannot be neglected. The interference affects the characteristics on the viewing angle of the device. We have performed 2D FDTD analysis for sine wave and Gaussian pulse incidence.

2.1 Optical simulation for the grating structure by sine wave incidence

The OLED device with the grating structure is shown in Figure 5(a). The settings of Perfect Matching Layer (PML) and Total Field and Scattering Field (TF/SF) for optical simulation are shown in Figure 5(b). The steady state field response for the sine wave incidence at 75 degrees is calculated.
In the syntax of Atlas, the scattering region in FDTD in the TF/SF setup is defined by “BEAM” statement as follows.

```
beam num=1 x.origin=0 y.origin=-0.5 angle=75 \   
fdtd fd.auto tm td.srate=4 sine \   
prop.length=10 big.index wavelength=0.45 \   
td.every=250 td.many=250 td.err=0.0025 dt=2.5e-18 \   
td.log=fdtdlog_test td.file=fdtd_test  fdtd.lum \   
FACET=nearfar_test scat.top=0.5 scat.bottom=0.2 \   
x.periodic angl.res=0.8
```

Simulated intensity of the electric field (Ez) is shown in Figure 6.

The incident light is reflected on the surface of gratings and the bottom electrode. Above the scattering region (y< -1.5), reflected light is separated from the incident light.

2.2 Optical simulation for the grating structure by Gaussian pulse

The device structure and the optical settings are the same as the previous section 2.1. The transient field response for the Gaussian pulse incidence at 75 degree is calculated. Using Gaussian pulse, the details of light propagation, reflection and scattering can be evaluated.

In the syntax of Atlas, Gaussian pulse setup is defined by “BEAM” statement as follows.

```
beam num=1 x.origin=0 y.origin=-0.5 angle=75 \   
fdtd fd.auto tm td.srate=4 pulse td.width=5e-16 \   
prop.length=10 big.index wavelength=0.45 \   
td.every=250 td.many=250 td.err=0.0025 \   
td.log=fdtdlog_test td.file=fdtd_test  fdtd.lum \   
FACET=nearfar_test scat.top=0.5 scat.bottom=0.2 \   
x.periodic angl.res=0.8
```

Simulated intensities of the electric field (Ez) are shown in Figure 7.

The incident light is propagated from the source on the top of the device, then reflected on the surface of the gratings and the bottom electrode. The details of the light propagation within the device can be precisely evaluated.

Conclusions

The newly implemented TMM Green’s function in Atlas has successfully simulated the output spectral power density from multiple dipole sources within the multi-layer OLED device for both coherent and incoherent conditions.

2D FDTD in Atlas simulated steady state and transient state field responses for oblique sine wave and Gaussian pulse incidence to the structure with periodic boundary condition. These newly implemented features are very useful for the optical modeling of OLEDs.
References


[7]. “Radiant: GUI-based Design Software for Performing Simulations of Optoelectronic Thin Film Devices Such as LED and OLED”, simulation Standard, Volume 25, Number 2, April-May-June 2015.
Deep Hole Etching Simulation for Advanced NAND Flash Memory

Introduction
The NAND Flash memory cell has been refined to reduce the bit cost, but the limit of its miniaturization has been reached due to the high electric field problem and the difficulty of lithography. On that account, three-dimensional stack cell structures have been adopted to achieve mass storage devices [1-3]. It has already been reported that the fabrication of 256Gbit NAND Flash memory with 48 stacked layers started on August in 2015 [4, 5]. For the fabrication of the stack structures, it is necessary to realize etching of deep holes. For examples, if using 30nm design rules and one layer thickness is 40nm, its depth becomes 1.92um. If the holes diameter is 100nm, its aspect ratio becomes 19.2. Then, in the next generation, the 512 Gbit flash memory cell will need the deep hole with the aspect ratio of 38.4 for 3.84um-depth. For investigating more suitable process conditions or optimum etched topography, accurate three-dimensional etching simulation is required, but it takes a very long time to simulate this deep hole etching accurately if using usual simulation methods like the Monte-Carlo method, because the aspect ratio of this deep hole is very large and therefore, the flux calculation effort of enhancing ions and neutral radical species is enormous for a reactive enhanced ion etching (RIE) model.

We have demonstrated the deep hole etching simulation with practical calculation times even for 3D, using the multi-dimensional process simulator Victory Process, which extensively and efficiently uses parallel processing. In this article, we show its simulation result with the ion enhanced chemical etching model and a good performance scaling of the parallel processing.

Model for the deep hole etching
Victory Process offers several etching models within its default open etching model library, from which we selected the Ion Enhanced Chemical Etching (IECE) model. As shown in the Figure 1, the IECE model assumes that the plasma consists of two types of species.

One is neutrals, which are uncharged thermal particles that chemically react with the surface.

- The other is ions, which are charged accelerated non-reacting particles that facilitate reactions of the surface material with the neutrals by removing reactions by-products from the surface, exposing it for chemical reactions.

Here, the final etch rate is written down as a function of the flux of neutrals and ions.

\[ R = (R_{\text{neutral}} f_{\text{neutral}} + R_{\text{ion}} f_{\text{ion}}) \theta_{ss} \]

It is assumed that \( R \) is the etch rate of the substrate, \( f_{\text{neutral}} \) is the partial neutral flux, \( f_{\text{ion}} \) is the partial ion flux, \( R_{\text{neutral}} \) is the neutral related etching rate, \( R_{\text{ion}} \) is the ion related etching rate, and \( \theta_{ss} \) is the surface coverage of molecules of substrate atoms and the neutral atoms in the steady state.

\[ \theta_{ss} = \left( j_{n} S_{0} / b \right) / \left( j_{n} S_{0} / b + \beta(E) j_{i} d_{\text{max covered}} \right) \]

Here, it is assumed that \( j_{n} \) is the local flux of the neutral atoms toward the surface, \( j_{i} \) is the local flux of the ion atoms toward the surface, \( S_{0} \) is the sticking coefficient of the neutral atoms on the clean substrate surface, \( b \) is the average number of the neutral atoms consumed by the substrate atoms, \( \beta(E) \) is the ion sputtering efficiency that depends on the ion energy, and \( d_{\text{max covered}} \) is the maximum desorption rate that corresponds to the full covered surface by the molecules of the neutral atoms and the substrate atoms.

The IECE model can be applied to any type of ion enhanced chemical etching as long as the enhancement is predominantly due to ions cleaning the substrate surface by removing the reaction by-products. In case of using the IECE model for 3D deep hole etching simulations, the most difficult problem is that the effort to calculate the neutral flux and the ion flux is enormous. In order to optimize simulation time, we have made a simple and high speed method for the calculation of the neutral flux based on the method of Kokkoris et al. [6]. Then, regarding the ion flux, we efficiently use parallel processing to keep the simulation time within practical limits.
Results and Discussion

At first, regarding the neutral flux, the dependence on the sticking coefficient is shown in Figures 2 (a) and (b). Figure 2 (a) is for a trench (2D like geometry) of width/depth=1um/4um, and Figure 2 (b) is for a circular hole of diameter/depth=1um/4um. Those results agree with those in the paper of Kokkoris et al. [6] very well. Having this method in place we can apply it also to large aspect ratio holes as shown in Figure 3. There, the neutral flux density for a deep hole with diameter/depth=0.1um/5um is presented. The calculation of any of these curves only takes one second. The above results demonstrate that the neutral flux can be calculated with high speed. Then, those curves are represented as a simpler function, and these function can be implemented as a C-model function in the open etching model library of Victory Process.

Next, we show the performance of the parallel processing for a simulation of a half hole etching using the IICE model in Figure 4. This result was obtained on Dell PowerEdge R820 (E5-4650, 32core, 128GB). The calculation times with 10, 20 and 30 threads are about 4, 7.5 and 10 times faster than that of one thread, respectively. The parallel processing is only applied to the ion flux calculation, and you can see a good performance comparable to the ideal performance curve.

Next, in order to investigate the dependence on the mesh resolution, the simple 3D etching simulation topography of a hole (diameter=0.1um, depth=3um) was simulated with the resolution=0.02um, 0.015um...
and 0.005um and the results were compared. Figure 5 shows a 3D view of the half hole etching simulation result with the resolution=0.015um. In Figures 6, which shows 2D cross-sections, the various simulation results are compared at various depth positions of the hole. The resolutions of 0.005um, 0.02um and 0.015um correspond to the red, green and blue lines, respectively. The etching topographies near Z=0.0um, 1.0um, 2.7um and the bottom are shown in Figures 6 (a), (b), (c) and (d), respectively. The green line of resolution=0.02 um has some discrepancies near Z=0.0um and the bottom with the red line of resolution=0.005um, but the blue line of resolution=0.015um agrees well with the red line of resolution=0.005um. We used resolution=0.015um for etching simulations of multiple deep holes.

Finally, we performed etching simulations of 4 and 5 holes with diameter/depth=0.1/4um using 10 threads on the server: Xeon x5690 6(HT12)*2, 12 threads, 140GB. Those simulation results are shown in Figure 7 and Figure 8. The left image (a) in those two figures shows the 3D solid view, while the right image (b) shows the cross
sectional view in the center of the simulation domain in y-direction. The calculations of 4 and 5 holes were finished in the practical times of 5.18 hours and 6.95 hours, respectively.

Conclusion

A 3D etching simulator with high speed and high accuracy is required, which can simulate the deep hole etching with the practical calculation time for development and fabrication of advanced NAND Flash memory. We have shown that Victory Process is capable of achieving these requirements.

Regarding the neutral flux, a high speed method was developed based on the method of Kokkoris et al., and then, a very higher speed for 3D was also realized using a simple calibrated C-model function in the open etching model library. Regarding the ion flux, which in this case dominates the calculation effort, we could show that it efficiently makes use of parallel processing. We demonstrated that Victory Process simulates etching of 4 and 5 deep holes with diameter/depth=0.1um/4um using the IECE model in a practical amount of time.

References

Performance Evaluation of a New Hybrid MPI-thread Parallelized Direct Solver

Background

The solution of linear systems lies in the core of any TCAD simulation. On any nonlinear step of the computation a linear system needs to be solved. The size and condition number of the matrices in these linear systems vary significantly depending on the specific type of TCAD simulation. So in order to achieve fast convergence it is required that the linear solver has good performance, good accuracy, can handle cases of ill-conditioned matrices, and it would be nice if the solver works well on any size linear system.

The two main types of linear system solvers are Direct and Iterative solvers. The pros and cons of the two types of solvers are respectively: Direct solvers are very accurate but can require large amounts of memory for large size problems for example 3D problems and their performance for such problems is usually not very good. Iterative solvers on the other hand are less accurate compared to direct solvers can diverge for linear systems with ill-conditioned matrices, but have very good performance and are designed to handle large size problems.

The question is can we have a direct solver that can handle large 3D problems, not have excessive memory requirements and have performance similar to an iterative solver. And the answer is yes if we split the large problem into smaller ones and design a parallel direct solver with several levels of parallelism.

New Hybrid MPI-thread Parallelized Direct Solver:

The new PAS solver part of the Silvaco suite of linear system solvers is a direct solver parallelized both with MPI and pthreads.

The large matrix coming from a large 3D problem can be divided into pieces by a graph partitioning tool and distributed to several machines by launching MPI processes on each machine. This will resolve the issue with memory.

On each machine since most modern machines have multiple cores multiple threads can work in parallel on the piece of the matrix assigned to that machine. So here we have two levels of parallelism. The parallel implementation of the PAS solver resolves the issue with performance. The PAS solver is orders of magnitude faster than older generation direct solvers and can be close with respect to speed to an iterative solver.

On smaller sized problems it can even be faster than iterative solvers. The biggest advantage of course is the accuracy. For problems for which high accuracy is required, the matrix of the linear system is ill-conditioned and the iterative method diverges or it takes a large number of iterations to get convergence, the PAS solver provides fast convergence of the nonlinear solver and can make the overall computation faster.

To illustrate the performance of the PAS solver we ran a 3D Victory Device simulation with a matrix size 392 298 on a machine with two Intel Xeon E5 Series CPUs each with 18 cores for a total of 36 cores.

On the same machine we launched MPI processes and the first column of Table 1 shows the number of processes launched.

The first row shows how many threads per MPI process we have used. In the table we recorded the total time of the simulation in seconds. To get the total number of threads used for the computation multiply the number of MPI processes and the number of threads per MPI process.

On every diagonal starting from the bottom left and going to the top right we have equal number of threads. Some of the slots in the table are not filled because the machine had only 36 cores so we did not run with for example 2 X 32 = 64 threads, or 4 x 16 = 64 threads, etc.

The graph on Figure 1 shows the data from the row of Table 1 starting with 1 MPI. This means that we have only one process launched and 1, 2, ..., 32 threads respectively are working in parallel on the whole problem.

<table>
<thead>
<tr>
<th>PAS</th>
<th>1 thr/MPI</th>
<th>2 thr/MPI</th>
<th>4 thr/MPI</th>
<th>8 thr/MPI</th>
<th>16 thr/MPI</th>
<th>32 thr/MPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 MPI</td>
<td>10451.58 s</td>
<td>5950.09 s</td>
<td>3607.05 s</td>
<td>2465.28 s</td>
<td>2098.07 s</td>
<td>1605.51 s</td>
</tr>
<tr>
<td>2 MPI</td>
<td>8133.70 s</td>
<td>4296.44 s</td>
<td>2913.44 s</td>
<td>2263.81 s</td>
<td>1827.38 s</td>
<td></td>
</tr>
<tr>
<td>4 MPI</td>
<td>5252.20 s</td>
<td>3424.69 s</td>
<td>2358.55 s</td>
<td>1656.67 s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8 MPI</td>
<td>3977.47 s</td>
<td>2719.87 s</td>
<td>1736.52 s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16 MPI</td>
<td>3347.99 s</td>
<td>2197.41 s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32 MPI</td>
<td>2903.98 s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Total time of a Victory Device simulation.
The amount of time for the simulation decreases significantly with the increase of the number of threads. The speedup with 32 threads is over 6.5 times. See Table 2 below. Another observation is that there is no saturation point which occurs in many solvers i.e. even though more threads are used the computation does not become faster it might even become slower.

For the PAS solver the more threads we use the faster the simulation.

Table 2 shows the speedup that we get with the PAS solver. The calculations are based on the timing information from Table 1. The speedup with 1 MPI process is graphed on Figure 2. The speedup is almost linear. The speedup increases in a similar fashion on every row of Table 2 so when you split the matrix you continue to get close to linear increase in the speedup.

If you look at the first column of Table 2 you can see that splitting the matrix into pieces and using only 1 thread for each piece is not such a good idea the increase of speedup is slower. This is due to the communication overhead caused by the transfer of data between the MPI processes.

As you increase the number of MPI processes the pieces per process become smaller and also having only 1 thread per piece makes the second level of parallelism nonexistent.

If you do decide to launch more than one MPI processes it seems that using 4 MPI processes and splitting the matrix in 4 pieces and then running 8 threads on each piece works very well also, the speedup is 6.3 for this case.

For this case the pieces of the matrix are large enough there are only 4 and for each piece you have enough resources - 8 threads per piece - so that the whole computation is close to optimal.

Next we will illustrate the performance of the PAS solver on more than one machines. The 3 machines that we used are identical with two Intel Xeon E5 Series CPUs each with 18 cores for a total of 36 cores per machine. On each machine we launched one MPI process.

We can see from the timing information in Table 3 that the best performance is achieved on 3 machines with 32 threads working in parallel on each of the 3 pieces in which the problem has been divided. The best time in this case is better than what we got on 1 machine so again we confirm that the more resources are used for solving the problem the better the time. The speedup that we got when we used 3 machines is 8.14 which is a very good result.

<table>
<thead>
<tr>
<th>PAS</th>
<th>1 thr/MPI</th>
<th>2 thr/MPI</th>
<th>4 thr/MPI</th>
<th>8 thr/MPI</th>
<th>16 thr/MPI</th>
<th>32 thr/MPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 MPI</td>
<td>1</td>
<td>1.76</td>
<td>2.9</td>
<td>4.24</td>
<td>4.98</td>
<td>6.5</td>
</tr>
<tr>
<td>2 MPI</td>
<td>1.3</td>
<td>2.43</td>
<td>3.59</td>
<td>4.6</td>
<td>5.72</td>
<td></td>
</tr>
<tr>
<td>4 MPI</td>
<td>1.99</td>
<td>3.05</td>
<td>4.43</td>
<td>6.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8 MPI</td>
<td>2.63</td>
<td>3.84</td>
<td>6.02</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16 MPI</td>
<td>3.12</td>
<td>4.76</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32 MPI</td>
<td>3.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Speedup of a Victory Device simulation calculated based on the timing information from Table 1.
Conclusion

We have added PAS a new hybrid MPI-thread parallelized direct solver to the Silvaco suite of linear system solvers. It complements our existing set of solvers and in particular serves as an alternative to our domain decomposition MPI parallelized iterative solver PAM for cases where we have convergence issues. The two solvers cover the large range of different types of linear systems that need to be solved in the TCAD simulations both in 2D and 3D.

The PAS solver is a modern parallel solver, very accurate and robust, with great performance and reasonable memory needs. It has the advantage of being able to run on several machines, and in parallel on each machine thus using their resources in an optimal manner.

<table>
<thead>
<tr>
<th>PAS</th>
<th>1 thr/MPI</th>
<th>2 thr/MPI</th>
<th>4 thr/MPI</th>
<th>8 thr/MPI</th>
<th>16 thr/MPI</th>
<th>32 thr/MPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 mach</td>
<td>10451.58 s</td>
<td>5950.09 s</td>
<td>3607.05 s</td>
<td>2465.28 s</td>
<td>2098.07 s</td>
<td>1605.51 s</td>
</tr>
<tr>
<td>2 mach</td>
<td>8249.34 s</td>
<td>4231.33 s</td>
<td>2815.28 s</td>
<td>1935.84 s</td>
<td>1592.76 s</td>
<td>1321.30 s</td>
</tr>
<tr>
<td>3 mach</td>
<td>7071.58 s</td>
<td>3893.42 s</td>
<td>2262.28 s</td>
<td>1717.47 s</td>
<td>1413.03 s</td>
<td>1283.02 s</td>
</tr>
</tbody>
</table>

Table 3. Timing information for PAS solver on more than one machines.
How can I create scalable devices using Victory Process?

Deckbuild supports variable substitution for both numerical and string variables using the SET statement and the $ symbol, thus allowing users to parameterize their input decks. The SET statement is used to generate a new variable and assign an initial value to it, e.g.,

```
SET scf=0.1
```

The value assigned to a variable can be numeric, string, boolean or a list. The variable holding the value is accessible by the $ symbol. Users can select any mix of masks, process parameters, device parameters, circuit parameters or tuning coefficients as input variables and then substitute variable values to investigate the output responses to input variables.

The possibility of input deck parametrization is considered very desirable, especially if the device under simulation is scalable and of circular shape. An example is an input deck for a nanowire transistor:

```
go victoryprocess
set scf=1
set LCHAN=20*$scf
set LACCE=5*$scf
set EPOX=1*$scf
set EPINV=1*$scf
set EPPOLY=5*$scf
set dm=10*$scf+$EPINV/2+$EPOX+$EPPOLY
init depth = $LCHAN polysilicon
gasheight=10 from="0",-$dm" to="$dm," flow.dim=3d dopant=arsenic
dopingvalue=le19
specifymaskpoly circle center="0,0" radius=10*$scf npoints=50 maskname="cir_a"
specifymaskpoly circle center="0,0" radius=10*$scf+$EPINV/2 npoints=50 maskname="cir_b"
specifymaskpoly circle center="0,0" radius=10*$scf+$EPINV/2+$EPOX npoints=50 maskname="cir_c"
specifymaskpoly circle center="0,0" radius=10*$scf+$EPINV/2+$EPOX+$EPPOLY npoints=50 maskname="cir_d"
line x loc=0 spac=1*$scf
line x loc=5*$scf spac=3*$scf
line y loc=-5*$scf spac=3*$scf
line y loc=0 spac=1*$scf
line y loc=5*$scf spac=3*$scf
line z loc=-$LACCE spac=$LACCE/5
line z loc=0 spac=$LCHAN/5
line z loc=$LCHAN spac=$LCHAN/5
etch polysilicon thickness=$LCHAN \ mask="cir_c" reverse max
deposit oxide thickness=0 max
etch oxide thickness=$LCHAN \ mask="cir_b" reverse max
deposit silicon thickness=$LACCE max
etch silicon thickness=$LACCE \ mask="cir_a" max
etch silicon thickness=$LACCE \ mask="cir_b" max
etch polysilicon thickness=$LCHAN \ mask="cir_d" max
export victory (conformal) structure=r1.str
```

Victory Process facilitates creation of a nanowire structure by allowing circular masks to be directly specified in the input deck with the SPECIFYMASKPOLY statement:

```
SPECIFYMASKPOLY CIRCLE CENTER="<x>, <y>" RADIUS=<Value> \ NPPOINTS=<Number> \ MASKNAME="circular_mask"
```

Using parametrization users can easily make the nanowire device longer or taller or even narrower or shorter. For example, users may use scf as a variable to store the value of RADIUS. The following figures show the construction of a nanowire transistor with different radii by means of variable substitution for RADIUS.

RADIUS=10  RADIUS=7  RADIUS=5  RADIUS=3

Hints, Tips and Solutions Archive

Check out our Web Page to see more details of this example plus an archive of previous Hints, Tips, and Solutions www.silvaco.com
JOIN THE WINNING TCAD TEAM

USA Headquarters:

Silvaco, Inc.
4701 Patrick Henry Drive, Bldg. 2
Santa Clara, CA 95054 USA

Phone: 408-567-1000
Fax: 408-496-6080

sales@silvaco.com
www.silvaco.com

Worldwide Offices:

Silvaco Japan
jpsales@silvaco.com

Silvaco Korea
krsales@silvaco.com

Silvaco Taiwan
twsales@silvaco.com

Silvaco Singapore
sgsales@silvaco.com

Silvaco Europe
eusales@silvaco.com