Expanding TCAD Simulations from Grid to Cloud

H. Demel, Z. Stanojević, and M. Karner
Global TCAD Solutions GmbH
Landhausgasse 4/1A, 1010 Wien, Austria
{h.demel|z.stanojevic|m.karner}@globaltcad.com

G. Rzepa and T. Grasser
Institute for Microelectronics, TU Wien
Gußhausstraße 27–29, 1040 Wien, Austria
{rzepa|grasser}@iue.tuwien.ac.at

***Abstract*—In this work, the distribution, execution and performance of TCAD simulations on grid and cloud systems are investigated. A module for distributed computing which can uniformly interface both grid and cloud computing systems has been implemented within GTS Framework. Automated allocation of resources for user jobs on a combined platform has been achieved. Traditional grid-computing systems are compared with cloud-based systems. Strategies for cost-effective allocation of cloud-resources are presented. The performance of a typical TCAD application run on a grid, in the cloud, and a hybrid system combining both are assessed.**

I. Introduction

Cloud computing has been heavily used for high traffic web-services for several years [1]. In that domain, resources need to be expanded to meet web-service's demands at any given time - otherwise the service will break down.

In the domain of TCAD, where grid-computing is pervasive, system stability is less of a concern. However, in case of time-critical jobs, or when grid resources are scarce, it may be desirable to add more resources to the system. Cloud computing provides a way to do so, tapping into a virtually unlimited resource-pool, but it has to be used with care to keep the expenses within reasonable bounds.

In this work, we compare traditional grid-computing systems with cloud-based systems, present strategies for cost-effective allocation of cloud-resources, and assess the performance of a typical TCAD application run on a grid, in the cloud, and a hybrid system combining both.

II. Cloud vs. Grid

A module for distributed computing was implemented within GTS Framework [2], which can uniformly interface both grid and cloud computing systems, and automatically allocates resources for user jobs. Figures 1 and 2 show the architecture of a grid computing system and a cloud computing system, respectively, as used by GTS Framework for distributed simulation jobs. In the grid case, execution hosts are submitted into the grid queue, which then perform a single simulation task when scheduled. In the case of the cloud, virtual machines (nodes) are created in the cloud and a execution host is started on each of them.

III. Allocation model

Most cloud computing providers use block based billing, typically by the hour [3], meaning that with the beginning of every new hour of node uptime a fee is charged. Starting a

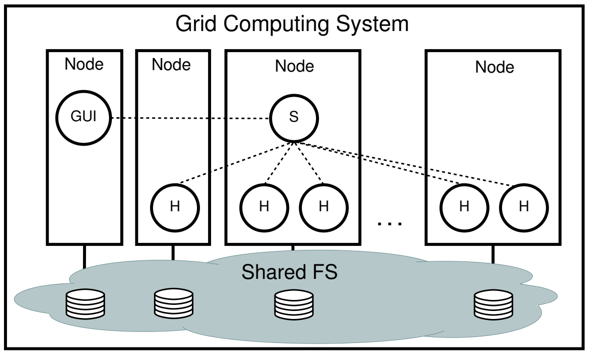


Figure 1. In a grid computing environment the Graphical User Interface (**GUI**) runs on a login node and submits a server instance (**S**) managing the simulation to the grid. GUI and server find each other by using a shared file system. The server submits simulation hosts (**H**) to the grid as needed.

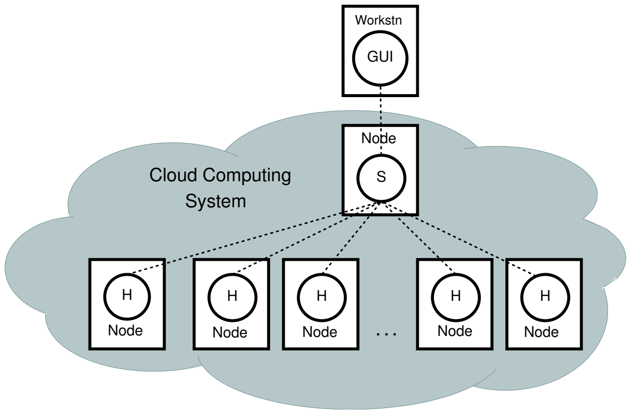


Figure 2. In a cloud computing environment the user workstation running the **GUI** is running outside of the cloud. The server instance (**S**) managing the simulation can run inside or outside the cloud and starts nodes that act as simulation hosts (**H**) via the cloud API as needed.

node for every simulation task would be wasteful if tasks are significantly shorter than the billing block time.

An algorithm is needed which minimizes the time until job completion while maintaining minimum cost-overhead. For an ideal solution, a-priori knowledge about the runtime of each task is required. In practice, however, task runtime cannot be known beforehand. The algorithm thus iteratively estimates

978-1-4673-7860-4/15/$31.00 ©2015 IEEE

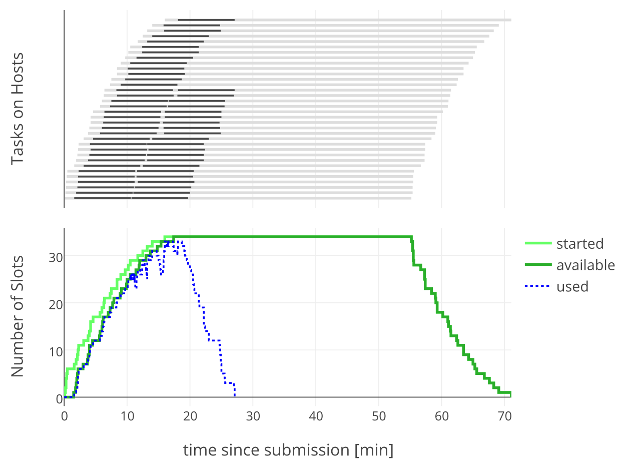


Figure 3. Resources used by naive algorithm, short tasks; Visualization of the uptime of computational nodes using the naive approach for 55 short tasks with 9 minutes of calculation time each and a simulated boot time of 90 seconds. Because of the delayed boot the number of started hosts is reduced to 33, which amount in costs for 33 machine hours. The minimum required amount would have been 10 machine hours only.

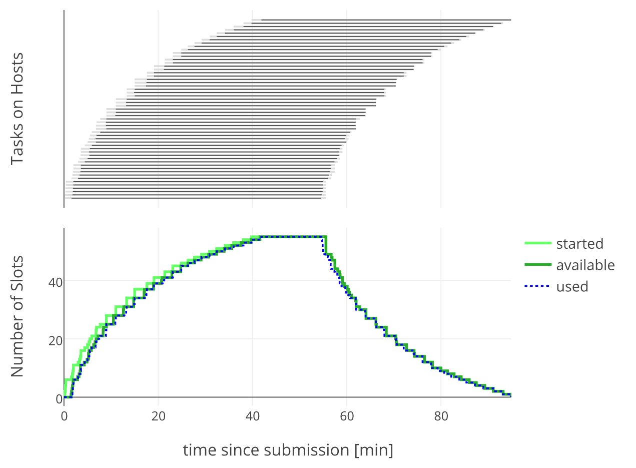


Figure 4. Resources used by naive algorithm, long tasks; Visualization of the uptime of computational nodes using the naive approach for 55 long tasks with 53 minutes of calculation time each and a simulated boot time of 90 seconds. Since the computational time is close to the billing block time of an hour, a machine should be started for each task. Using the naive approach the last computational node is started after 39 minutes leading to an unnecessary extension of the total runtime.

the number of required nodes, requests these in the cloud, and waits until the requested nodes have booted. The procedure is then repeated with an updated estimate for the number of required nodes.

A naive approach at estimating the number of required nodes would be to use a fixed percentage of pending tasks, e.g. $n_{\text{req}} = n_{\text{pend}} \times 10\%$. This approach is effective when dealing with short tasks, but produces significant cost overhead, as can be seen in Fig. 3.

Reducing the set percentage will not reduce the overhead significantly but increases the total runtime, especially for many nodes and longer task time. The increase in runtime is due to the exponential convergence behavior of the naive algorithm and the resulting long delay in the start up of the last computational node (cf. Fig. 4).

A refined, heuristic approach computes an expected runtime t_{exp} for each task, based on data collected from completed and running tasks. The number of requested nodes is

$$n_{\text{req}} = \Big[\sum_{\text{task}} (t_{\text{exp}} - t_{\text{run}}) - \sum_{\text{node}} (t_{\text{block}} - t_{\text{up}})\Big] / 50\,\text{min} \qquad (1)$$

where t_{run} is the already spent runtime of a task, and $t_{\text{block}} - t_{\text{up}}$ is the time until a node enters the next billing block. This approach reduces cost overhead for short tasks, but increases overall runtime due to fragmentation.

To mitigate this, a fast and simple scheduling algorithm is used in conjunction with the node-requesting heuristic to reduce fragmentation. This reduces the overall runtime while maintaining cost at a minimum. The start-up and scheduling processes are visualized in Fig. 5.

IV. Results and Discussion

The autonomous scaling capabilities can also be used in a *hybrid* setup where an existing grid computing system is extended by scaling out to the cloud when too little resources are available on the grid. We tested the hybrid system using a reliability/variability simulation of a 20 nm FinFET. Due to the small dimensions of the device, the assumption that the doping atoms are uniformly distributed and form a continuum breaks down [4]. The position of each individual doping atom has a decisive influence on the transistor's properties. For that reason, based on the transistor design, a multitude of virtual devices is generated, each with a different random distribution of discrete dopants. One of these generated devices is shown

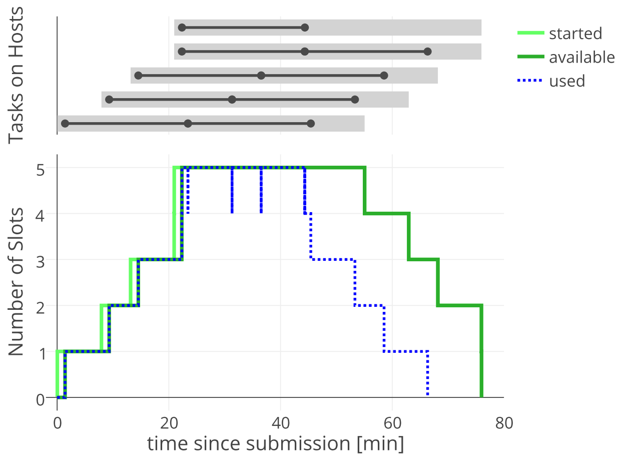


Figure 5. Visualization of the start-up of hosts based on the heuristic approach with a scheduler; after the submission of nine tasks, one host is started immediately. As soon as the first task exceeds a runtime of six minutes a second host is added since it is not possible to execute all tasks on one host within an hour. If the tasks were shorter than 12 minutes, ideal scheduling would be to execute four tasks on one host and five tasks on the other. After 12 minutes this is not possible within an hour anymore and further nodes are added.

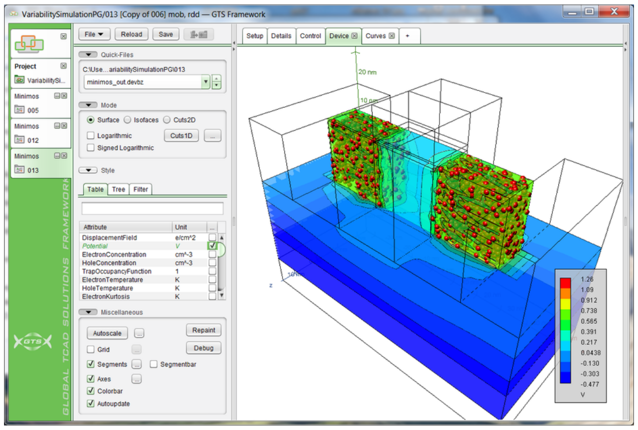
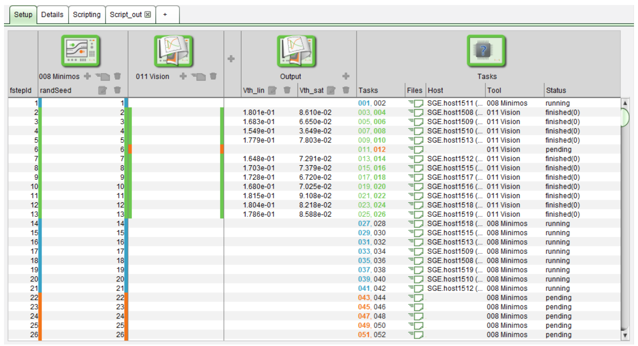


Figure 6. Virtual device with random discrete dopants shown in the GTS simulation framework



fstepId	randSeed	Vth_lin	Vth_sat	Tasks	Files	Host	Tool	Status
1	1			001, 002		SGE.host1511 (...	008 Minimos	running
2	2	1.801e-01	8.610e-02	003, 004		SGE.host1508 (...	011 Vision	finished(0)
3	3	1.683e-01	6.650e-02	005, 006		SGE.host1509 (...	011 Vision	finished(0)
4	4	1.549e-01	3.649e-02	007, 008		SGE.host1510 (...	011 Vision	finished(0)
5	5	1.779e-01	7.803e-02	009, 010		SGE.host1513 (...	011 Vision	finished(0)
6	6			011, 012			011 Vision	pending
7	7	1.648e-01	7.291e-02	013, 014		SGE.host1512 (...	011 Vision	finished(0)
8	8	1.703e-01	7.379e-02	015, 016		SGE.host1515 (...	011 Vision	finished(0)
9	9	1.728e-01	6.720e-02	017, 018		SGE.host1517 (...	011 Vision	finished(0)
10	10	1.680e-01	7.025e-02	019, 020		SGE.host1516 (...	011 Vision	finished(0)
11	11	1.815e-01	9.108e-02	021, 022		SGE.host1516 (...	011 Vision	finished(0)
12	12	1.804e-01	8.218e-02	023, 024		SGE.host1518 (...	011 Vision	finished(0)
13	13	1.786e-01	8.588e-02	025, 026		SGE.host1519 (...	011 Vision	finished(0)
14	14			027, 028		SGE.host1518 (...	008 Minimos	running
15	15			029, 030		SGE.host1515 (...	008 Minimos	running
16	16			031, 032		SGE.host1513 (...	008 Minimos	running
17	17			033, 034		SGE.host1509 (...	008 Minimos	running
18	18			035, 036		SGE.host1508 (...	008 Minimos	running
19	19			037, 038		SGE.host1519 (...	008 Minimos	running
20	20			039, 040		SGE.host1510 (...	008 Minimos	running
21	21			041, 042		SGE.host1512 (...	008 Minimos	running
22	22			043, 044			008 Minimos	pending
23	23			045, 046			008 Minimos	pending
24	24			047, 048			008 Minimos	pending
25	25			049, 050			008 Minimos	pending
26	26			051, 052			008 Minimos	pending

Figure 7. Script tool view showing the running simulation setup; the Structure tool is initialized with a varying seed for the random dopant distribution; the created device is passed on to a Minimos-NT simulation; the Vision tool is used to postprocess the results and determine $V_{\mathrm{th,lin}}$ and $V_{\mathrm{th,sat}}$; the current progress of the tasks is color-coded

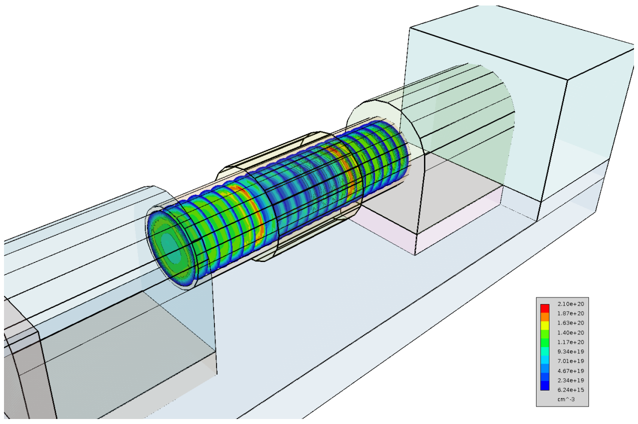


Figure 8. Full 3D device simulation of a silicon nanowire; The subband-Boltzmann transport equation is solved on slices along the channel region to determine an effective mobility. The GTS job server takes care of multi-host parallelization of the subband and scattering rate calculation (see Fig. 9).

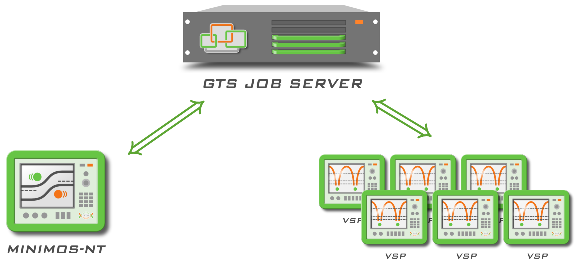


Figure 9. Simulators can make use of the job server to demand hosts for sub-processes: The GTS job server starts a Minimos-NT job. The device simulator requests VSP jobs from the job server to calculate the mobility on device cuts. The job server distributes the tasks and reports the VSP output back to Minimos-NT.

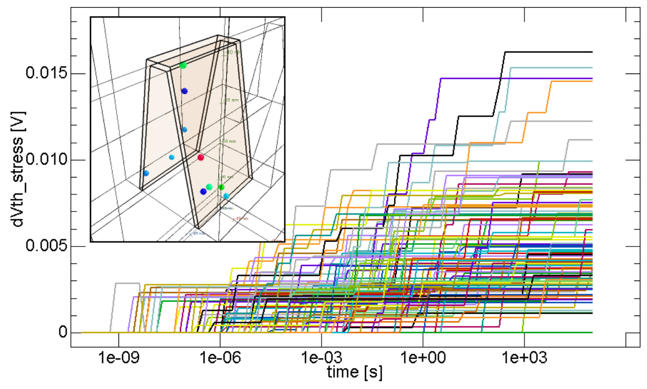
in Fig. 6. The simulation tasks consists of determining the current/voltage characteristics for the linear and the saturated operation regime of the transistor. The threshold voltages for both regimes, $V_{\mathrm{th,lin}}$ and $V_{\mathrm{th,sat}}$, are determined for each doping configuration. An exemplary view of such a simulation is shown in Fig. 7. The GTS job server allows for an easy task setup and shows the job progress and postprocessing results in a single view.

Another example is the full 3D device simulation of a silicon nanowire (Figure 8). The subband-Boltzmann transport equation is solved on slices along the channel region to determine an effective mobility [5]. The GTS job server takes care of multi-host parallelization of the subband and scattering rate calculation (see Fig. 9).

The shift of the threshold voltage is an important key figure for the transistor's performance in an integrated circuit. Of special interest is the BTI-induced V_{th} shift, where a small change in oxide trap distribution can lead to considerable changes in V_{th} under device-stress conditions [6]. Large samples of trap/dopant configurations need to be investigated to achieve good statistics on the device reliability.

A typical result of such a reliability/variability simulation using a sample size of 480 can be seen in Fig. 10. Each of the individual simulation tasks run on a grid took between 13.2 min and 19.3 min with an average of 17.6 min. The distribution of the computation times of the simulation tasks is shown in Fig. 11. The difference in simulation runtime is due to the varying convergence of the equation system, that depends strongly on the random dopant configuration. The same set on tasks run on a cloud system took between 21.7 min and 28.5 min with an average of 24.7 min. This is due to the lower floating point performance of typical cloud machines compared to a computing grid specifically built for TCAD simulation tasks.

The measurements of the time until all computations are finished and the associated costs (Figure 12) shows that using the naive algorithm leads to multiples of the necessary cost for short computations. The heuristic algorithm based on the numeric formula drastically reduces the cost, but induces longer run times. The iterative algorithm including a scheduler always causes as much or even less costs than the algorithm



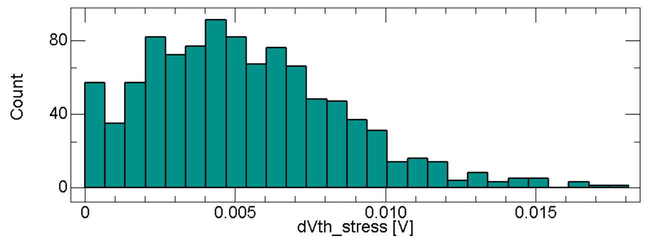


Figure 10. BTI-induced V_{th}-shift in an ensemble of n-FinFETs with randomly positioned oxide traps (top); ΔV_{th}-distribution after a stress time of 10^5 s (bottom)

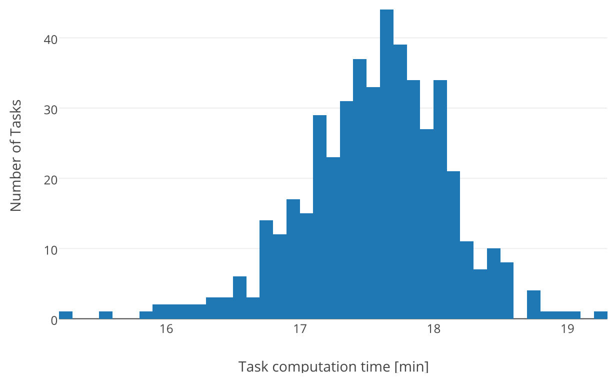


Figure 11. Histogram of the computation time of the simulation tasks for different random dopant configurations run on a grid

based on the heuristic-only approach and provides much faster run times, for task run times above half the billing block time (i.e. half an hour) even faster than the naive algorithm.

In this setting we found that the hybrid system delivers results faster than pure cloud or pure grid systems and costs less than the pure cloud system as summarized in Table I.

V. Conclusion

A newly developed algorithm combining an heuristic formula with a scheduling algorithm for determining the number of nodes to start in the cloud proved to be able to keep the expenses of cloud usage as low as possible and speed up the time until results can be obtained considerably. In a hybrid solution the costs can be reduced even further by only scaling

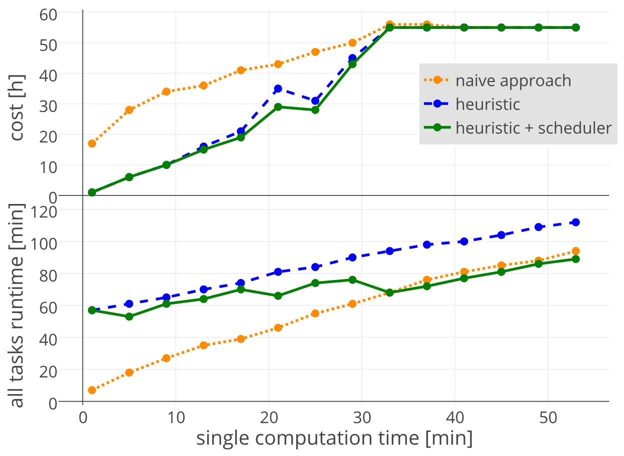


Figure 12. Comparison of three algorithms to determine the number of nodes to start in the cloud for 55 computational tasks; the heuristic algorithm with scheduler needs less resources than the naive approach for short simulations and delivers results more quickly for long running simulations by starting up more nodes earlier.

Table I
The hybrid setup delivers the results fastest and costs less than the cloud-only setup assuming that grid usage is cheaper than cloud usage.

	runtime	cost
grid	108.1 minutes	17.7 grid machine hours (8 tasks per machine)
cloud	69.3 minutes	72 cloud instance hours (4 tasks per instance)
hybrid	62.4 minutes	6.2 grid machine hours 43 cloud instance hours

out during load peaks. We conclude that the hybrid system combining cloud and grid is a feasible and economic approach to handle time-critical large-scale TCAD simulations.

References

[1] Michael Armbrust, Armando Fox, R. Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, G. Lee, D. A. Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia, “Above the Clouds: A Berkeley View of Cloud Computing,” Feb. 2009.

[2] “GTS Framework, http://www.globaltcad.com/framework.” [Online]. Available: http://www.globaltcad.com/framework

[3] P. Marshall, H. Tufo, K. Keahey, D. LaBissoniere, and M. Woitaszek, “A Large-Scale Elastic Environment for Scientific Computing,” in *Software and Data Technologies*, ser. Communications in Computer and Information Science, J. Cordeiro, S. Hammoudi, and M. van Sinderen, Eds. Springer Berlin Heidelberg, 2013, vol. 411, pp. 112–126.

[4] N. Sano and M. Tomizawa, “Random dopant model for three-dimensional drift-diffusion simulations in metal-oxide-semiconductor field-effect-transistors,” *Applied Physics Letters*, vol. 79, no. 14, pp. 2267–2269, 2001. [Online]. Available: http://scitation.aip.org/content/aip/journal/apl/79/14/10.1063/1.1406980

[5] M. Karner, Z. Stanojević, F. Mitterbauer, C. Kernstock, and H. Demel, “Bringing Physics to Device Design – a Fast and Predictive Device Simulation Framework,” in *2015 Silicon Nanoelectronics Workshop (SNW)*, 2015, pp. 75–76.

[6] Hui-Wen Cheng, Fu-Hai Li, Ming-Hung Han, Chun-Yen Yiu, Chia-Hui Yu, Kuo-Fu Lee, and Yiming Li, “3D device simulation of work function and interface trap fluctuations on high-K / metal gate devices,” in *Electron Devices Meeting (IEDM), 2010 IEEE International*, Dec. 2010, pp. 15.6.1–15.6.4.