CSim: A MOS Switch-Level Simulator

Fangzhou Wang, Sandeep K. Gupta

Computer Engineering Technical Report Number CENG-2016-03

Ming Hsieh Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, California 90089

December 2016

CSim: A MOS Switch-Level Simulator

Fangzhou Wang
Ming Hsieh Department of Electrical Engineering
University of Southern California
Los Angeles, USA 90089
fangzhow@usc.edu

Sandeep K. Gupta
Ming Hsieh Department of Electrical Engineering
University of Southern California
Los Angeles, USA 90089
sandeep@usc.edu

ABSTRACT

We develop a new switch-level simulator, CSim, for MOS transistor cells. CSim simulates cells without circuit parameters, such as the sizing of transistors. To tackle the challenges caused by the lack of circuit parameters, a new transistor-level levelization algorithm is developed. SPICE simulator is used as reference simulator, CSim is accurate in a sense that the simulation results for a cell are consistent with SPICE simulation results for any transistor sizing configurations. The program was tested on all possible cells up to 4 transistors and was able to report accurate results for all levelizable cells.

1. INTRODUCTION

A slowdown in benefits of CMOS scaling motivates researchers to explore new materials and physical phenomena to develop new design alternatives. While it took us more than three decades to develop a design methodology based on modeling, exploration, and optimization for the CMOS technology before we can fully harness its benefits, the development cycle for post-CMOS alternatives needs to be far more compressed. Hence, it is imperative to rapidly exploit promising post-CMOS technologies. We need to discover a rich set of design primitives, or cell libraries without being heavily influenced by CMOS designs. Our objective is to develop a completely new approach based on computer-assisted, data-driven, and yet human-in-loop systematic explorations to enable fast discoveries that help rapidly harness the benefits of emerging technologies. Our proposed approach is significantly beyond traditional CAD, where algorithmic innovations are used to harness existing knowledgebase. What we proposed is a computer-aided approach that not only precedes the cells, principles, and methods, it actually helps in their discovery. Hence, to distinguish from CAD as we know it, we call what we propose to develop as Computer-Aided Discovery, or CADis for short. CADis uses a paradigm of successive-refinement of computer-aided tools for rapid discovery, with humans-in-loop being assisted by the tools and the data generated by the tools to rapidly make refinements and discoveries. Figure 1 shows a top-level view of CADis. It consists of three parts: (1) cell enumerator, (2) cell simulator, and (3) cell selector. A cell enumerator generates all possible configurations for a given number of components, a cell simulator is designed to generate accurate results based on the limited cell configuration information provided by the cell enumerator, a cell selector selects promising cells based on certain criteria.

We developed our first version of CADis for MOS to test

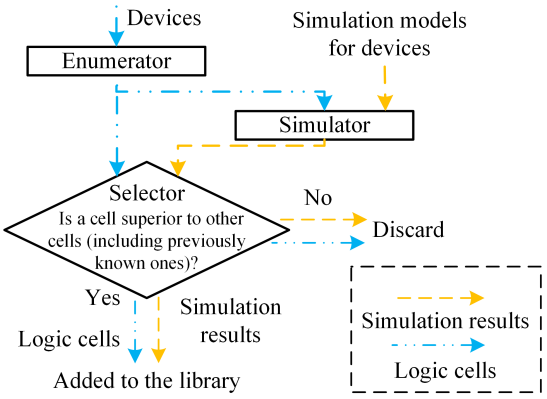

Figure 1: A top-level view of CADis

our proposed framework. An enumerator is developed to enumerate all possible distinct cell configurations. Though the complexity for topology enumeration is high, it is still manageable for small number of transistors. However, enumerating all possible transistor sizings would create astronomical number of cells that are hardly manageable. Thus, our cell enumerator only enumerates cell topology configurations. A cell simulator is used to compute the boolean function implemented by the enumerated cells. However, detailed circuit parameters are required by the existing simulators such as SPICE[1]. With the limited information provided by our enumerator, we decided to use switch-level simulators. The existing switch-level simulators such as IRSIM[2] would also require transistor sizing information to compute the strengths of transistors. Thus, a special switch-level simulator needs to be implemented.

In this report, we develop a special cell simulator to analyze astronomical numbers of cells and report accurate results.

2. MAIN CHALLENGES

A specially designed simulator enables us to analyze the very large number of cell configurations generated by our cell enumerator. *Our simulator is required to provide accurate simulation results based on the limited information provided in the cell configuration.* We use SPICE simulator as our reference simulator. That is, if the simulation results provided by our simulator are consistent with the SPICE simulation results, then the simulation result of our simulator is accurate.

Since we do not have the information about transistor sizes, we do not know the value of important circuit-level parasitics, such as capacitances of transistor gates and diffusion regions and the above consistency requirement must be satisfied for all possible combinations of transistor sizes. This is also one of the reasons for why we cannot use SPICE simulator to analyze cells generated by our cell enumerator. Without the sizing information, we cannot compute timing values and face many challenges. First, without timing information, only steady-state logic response can be computed. Second, if steady-state responses vary depending on the precise timing of various transients, then our simulator cannot report a specific steady-state response.

In addition to the limited information, another primary challenge for the simulator is the *bidirectionality* of the transistor channel. Every transistor channel is bidirectional with the only exception being when one of the diffusion regions of the transistor is connected to VDD, GND, or a primary input (each of these are assumed to be stronger than any internal node). This requires that (1) if an internal circuit node (including the output node) is connected to the diffusion region of one or more transistor, then the value at the diffusion region cannot be computed until the gate values are known for all these transistors, and (2) if two internal nodes n_1 and n_2 are connected to each other through a transistor channel, then the values of n_1 and n_2 must be computed at the same time.

Hence, we need to implement a special simulator that can perform simulation using only transistor-level netlist information and can tackle the challenges of transistor channels without assuming any transistor sizes.

3. MOS CELL SIMULATOR

In order to tackle the challenges of limited information and bidirectionality of the channel of a transistor, we design a new transistor-level levelization algorithm. This new levelization approach guarantees the correctness of our simulation results by identifying the subset of cells for which our cell simulator can accurately compute the steady-state response. If a cell is levelizable, then (1) the level of diffusion region of a transistor should always be at least one level greater than the gate of the transistor, and (2) for any transistor, both of its diffusion region should be at the same level. The only exception for these is that one of the diffusion region of the transistor is connected to VDD, GND, or primary input.

Our new levelization algorithm is shown in Algorithm 1. All the cell configurations that can be levelized by this algorithm hold the following crucial properties which enable our simulator to compute the steady-state response without the information about transistor sizes, hence without any information about transistor strengths and delays. Our simulation proceeds level by level, so it is guaranteed to provide correct steady-state response, if the cell can be levelized.

LEMMA 1. *For a given transistor, each of its diffusion regions will always be assigned a level that is at least one level higher than the level of the transistor's gate. The only exception are cases where the node that contains the diffusion region also contains a cell input, VDD, or GND.*

Proof: If the node that contains the diffusion region also contains a cell input, VDD, or GND, then that diffusion region will be assigned as $level = 0$. Otherwise, the levelization algorithm will assign level $j+1$ to the diffusion region if the gate is at level j. Since the levelization algorithm eventually chooses the highest level while updating the level of an internal node or the output node, the level of a diffusion region will remain $j + 1$ or increase. Hence, the level assigned to each diffusion region of a transistor will always be at least one level higher than the level of its gate (except for the above special case).

Algorithm 1: An outline of levelization algorithm

```
1  Phase - Initialization
2  Initialize level=undefined to every terminal of every
     transistor and every node
3  Pass - 0
4  For every node that contains a cell input, VDD, or
     GND, assign level=0 to every gate/diffusion region in
     the set and assign level=0 to this node.
5  Pass - i
6  repeat
7  |  foreach transistor gate that is assigned a new level
   |    j in the previous pass do
8  |  |  Assign level j+1 to each of its diffusion regions,
   |  |    and then assign the nodes that contains the
   |  |    diffusion regions a level that is maximum of the
   |  |    diffusion regions in the set. Update all other
   |  |    nodes that have channel connections to this
   |  |    node with the same level of this node. Update
   |  |    all the transistor gates that are contained in
   |  |    these modified nodes.
9  |  end
10 until until no change is made during the iteration or i
     is greater than the number of transistors in the cell;
11 Phase - Check
12 If all nodes in the configuration have been assigned a
     level and the maximum count is not reached, then the
     levelization is completed successfully, otherwise, the
     levelization fails.
```

LEMMA 2. *If a node N_d contains a diffusion region of transistor t and a node N_g contains the gate terminal of transistor t, then N_d is at least one level higher than N_g, except the case in which N_d also contains a cell input, VDD, or GND.*

Proof: By Lemma 1, each diffusion region of transistor t is always at least one level high than the gate terminal of t. According to our levelization algorithm, a node is assigned a level that is the maximum of the diffusion regions in the set. Hence, N_d is at least one level higher that N_g.

LEMMA 3. *Within a node, the level of any gate terminal that belongs to the node is the same as the level of the diffusion region which has the highest level.*

Proof: In our levelization algorithm, a gate terminal is assigned a level that is the maximum of the diffusion regions in the set.

LEMMA 4. *If a node N_g contains the gate terminal of a transistor t, then its value cannot be affected by a node N_d which contains one of the diffusion regions of t.*

Proof: When processing the nodes at level j, the simulation algorithm would not have updated the value of the

nodes at level $j+1$ or higher. If N_g can be affected by N_d, then N_g should have a level that is higher than or equal to N_d. However, by Lemma 2, N_d is at least one level higher than N_g, hence the value of N_g cannot be affected by N_d.

LEMMA 5. *For any transistor, the node containing its diffusion regions are at the same level. The only exception is that one of the diffusion region is connected VDD, GND, or primary input.*

Proof: In our levelization algorithm, all nodes that have channel connections with each other are updated to the same level.

THEOREM 6. *Our simulation algorithm provides correct steady-state respond if the cell can be levelized.*

Proof: Our simulation proceeds level by level, within each level it first gathers all the values of the nodes in that level and then computes the steady-state values for those nodes. So it first takes all the nodes that are at level 0 (which are VDD, GND, and Inputs), updates their values, and then moves to the transistors whose gates are known and updates the status of these transistors. Hence, the simulation process would update all level 1 nodes. If they are at level 1, then by Lemma 2, the status is known for all the transistors that are driving them. According to the levelization algorithm, all the other nodes that can affect its value are either from the same level or connected to VDD, GND, or input, so the values at the nodes at level 1 can be calculated. By Lemma 3 and 4, diffusion nodes can only affect either those diffusion nodes that are at the same level or transistor gates that are driven by these. Hence, as the simulation proceeds from low level to high level it eventually computes the steady-state response at the output of the circuit.

When we started to build the simulator, we only granted ourselves to the switch-level model of the transistor without any other detailed device characteristics. Since at the beginning of our exploration, the number of transistors being enumerated is small, we were able to exhaust all sizing options for a small portion of the generated cell so that SPICE simulation can be run. After running simulations using our simulator for small cells, we verified the simulation results with the SPICE results. There were mismatches between the simulation results for several cells. We then further investigated those cells and found out that the mismatches were caused by signals passing via cascaded transistors. This lead to the discovery of weak signal property of the MOS devices. Then we expanded the value set of our simulator to not only include strong logic 1 and strong logic 0, but also include weak logic 1 (denoted as i) and weak logic 0 (denoted as o). A weak logic 1 will be generated if a strong logic 1 is passed through a conducting NMOS, a weak logic 1 will become unknown if it is passed through a conducting NMOS which is turned on by weak logic 1. Similar property holds for PMOS devices. By enhancing our simulator with the notion of weak signals, all the simulation results generated by CSim match with the SPICE simulation results.

Extensions implemented: With a value set which contains strong logic 0, strong logic 1, weak logic 0 (o), weak logic 1 (i), and U, where U denotes *unknown*, our simulator can be accurate. However, in many cases this would produce too many U values at the output. By using special composite values such as R to denote *ratioed* value and Z to denote $high-impedance$ value, the precision of our simulator is improved while guaranteeing correctness. If the gate of a transistor is R then the value of the two diffusion regions will become R. If the gate of a transistor is U or Z, then the two diffusion regions of the transistor will become U. As shown in Fig. 2, if both logic 0 and logic 1 are affecting the value of a node through channels of conducting transistors or R is affecting the node through a conducting channel, then R will be generated for that node. If U is affecting a node through a conducting channel, then the value of that node will become U. Z can be affected by any other value, so all the nodes are initialized as Z.

V_y \ V_x	0	1	o	i	R	Z	U
0	0	R	0	R	R	0	U
1	R	1	R	1	R	1	U
o	0	R	o	R	R	o	U
i	R	1	R	1	R	i	U
R	R	R	R	R	R	R	U
Z	0	1	o	i	R	Z	U
U	U	U	U	U	U	U	U

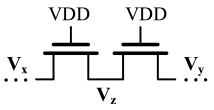

Figure 2: Value at node Z given value of X and Y when both transistors are conducting.

4. CONCLUSION

We have developed a new switch-level simulator for MOS technology to analyze the very large number of cell configurations generated by our cell enumerator. Since the cell configurations only have topology information (i.e., circuit parameters such as transistor sizing are not available), a new transistor-level levelization algorithm was developed to tackle the challenges of limited information and bidirectionality of the channel of a transistor.

Our simulator was able to compute accurate steady state respond for all cells that can be levelized by the levelization algorithm. SPICE simulations were performed on the same set of cells with a set of extreme sizing configurations. All simulation results computed by CSim match with the SPICE simulation results in a sense that when CSim reports *logic* 0 or *logic* 1, the result is consistent for any sizing for SPICE simulation.

Since our existing levelization algorithm levelizes cells without assuming any input pattern, cells that can be levelized under certain input patterns are excluded by the algorithm. In addition to this, even though some cyclic cells cannot be levelized, the steady state respond can still be computed (e.g., a cell consists of two back to back inverters). We are enhancing our levelization algorithm and our simulator to be able to simulate cyclic cells with certain characteristics and cells that can be levelized under certain input patterns to expand the scope of CSim.

5. REFERENCES

[1] K. Kundert. *The Designer's Guide to Spice and Spectre®*. Springer Science & Business Media, 2006.

[2] A. Salz and M. Horowitz. Irsim: An incremental MOS switch-level simulator. In *Design Automation, 1989. 26th Conference on*, pages 173–178. IEEE, 1989.