PLODE: Precise Logic and Delay Simulator for Structural Verilog

Gokce Nur Erer, Alp Arslan Bayrakci

Gebze Technical University
gokce.erer@gmail.com, abayrakci@gtu.edu.tr

Abstract

Spice simulator is famous with its precision and preferred as a final stage verification tool. However, it needs a different circuit description than Verilog, and also its direct results are not meaningful in terms of speed and accuracy without further analysis. Commercial tools are proprietary and using them for conversion to Spice deck and analysis of the Spice results is costly in terms of both finance and effort. Precise Logic and Delay (PLODE) is a novel, research oriented, open source tool that can convert structural Verilog to Spice deck, perform simulations at the desired process technology node, analyze and visualize the results in terms of delay and functional accuracy. In order to help path delay based research, it can also extract a path from the circuit and simulate it without the need for a sensitizing input pair. The tool is tested and verified on ten combinational and two sequential circuits.

1. Introduction

Spice transistor level circuit simulator [1] is a widely accepted precise tool for circuit simulations. More improved, GPU accelerated versions are produced by leading EDA companies like Synopsys and Cadence. Also, having the open source versions like `ngspice` [2] makes it an essential component for academic research, where it is very hard, expensive and time consuming to work on actual manufactured chips. For a wide range of research areas like timing analysis, hardware security, device modeling, delay and power based methods, etc., it is indispensable especially for final verification purposes as widely done accompanying Monte Carlo technique.

The commercial CAD tools have embedded software that can perform complex tasks including different levels of abstraction and different stages like design, simulation and layout extraction. Yet, they are proprietary and can not be modified, need profession and a full process design kit to be usable. From an academic research perspective, they may not be compulsory as most of their tasks may not be required for a particular research problem. The well-known Modelsim [3] simulator can be obtained free of charge accompanying Intel FPGAs with design size limitations. On the other hand Logisim [4] is an open source tool that enables graphical circuit design and can simulate the design for functional accuracy similar to Modelsim. But both Modelsim and Logisim do not perform precise transistor level simulations revealing the actual behavior of the circuit in terms of delay and functionality.

The gEDA [5] project is initiated to fill the gap in free EDA software and it contains different tools for schematic capture, netlist generation and utilizes Spice for simulation. However, it is PCB oriented and does not analyze the Spice simulation results either. VIS [6] is an open source circuit synthesis and verification tool, developed by University of California, Berkeley. But simulation is only for proof of concept similar to Modelsim. Yosys [7] and Odin [8] are open source Verilog synthesis tools, focused solely on behavioral Verilog synthesis to netlist. However, these tools do not perform a comprehensive conversion from structural Verilog to Spice, analysis of Spice simulation outcomes for the extraction of all required results like realistic output delays and functional accuracy.

In this paper, we propose Precise Logic and Delay (PLODE) simulator: a research-oriented, free, open source tool that can perform the conversion, simulation and analysis of digital circuits written in structural Verilog (Fig. 1). It accepts both combinational and sequential circuits. Supports hierarchical Verilog with nested module instantiations. It can flatten or preserve the hierarchy. The simulation part is handled by `ngspice`, therefore the limitations (like simulation duration) and the precision of PLODE, are not different from Spice. PLODE performs analysis on Spice simulations to deduce meaningful results for output delays and logic values important for speed and accuracy of the design. In order to help path delay based research, PLODE can also extract a given path from the circuit with all its side gates and simulate this single path to compute its path delay. We test the tool on ISCAS'85 [9] test circuits, on a 32 bit adder written in hierarchical Verilog, and on two ISCAS'89 [10] circuits to show its ability to simulate the sequential circuits. Same Verilog circuits are given to Modelsim for verifying the PLODE logic results.

The overview and capabilities of PLODE, the conversion from Verilog to Spice, the simulation and analysis stages as well as the single path mode of the tool and its GUI are explained in Section 2. Section 3 gives the details of the experimental setup and presents the results.

2. PLODE

2.1. Overview

As Fig. 1 shows, PLODE takes the circuit description in structural Verilog, the input vectors and expected output vectors as text files, the SCL cells as Spice subcircuits, the technology file for the target process and Spice options like temperature, reltol, etc. as its inputs.

The conversion of Verilog modules into Spice Decks is explained in Section 2.2. The simulation stage consisting of reading the input vectors and supplying them to the circuit and meanwhile collecting all results is in Section 2.3. Using the results of the simulation stage PLODE performs analysis to extract all output delays as well as the logic values for each input. This is called analysis stage and explained in Section 2.4. Path extraction for single path delay computation is presented in Section 2.5. Lastly, Section 2.6 presents the GUI of PLODE.

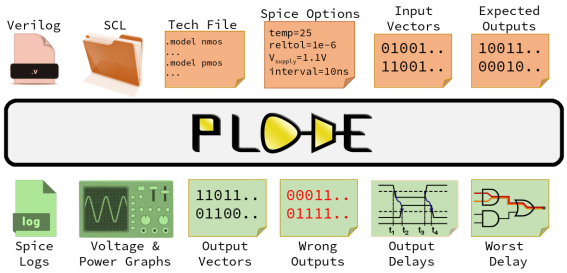

Figure 1. Overview of PLODE (top:inputs, bottom:outputs)

2.2. Conversion

PLODE converts the Verilog circuit into a graph structure before generating the Spice deck. In this graph structure, each logic cell, interconnect, primary input and primary output are represented as a node. Fig. 2 shows the graph for a `full_adder`.

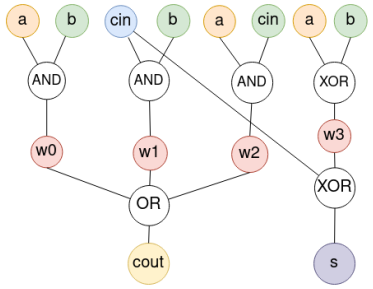

Figure 2. Graph constructed for `full_adder`

For hierarchical Verilog with module instantiations, the tool provides two options for Verilog to Spice conversion: *Graph-joining* and *Spice-subcircuits*. If the Graph-joining option is selected, the hierarchical circuit is flattened into one level and represented by one graph utilizing only the standard cells in the SCL, similar to Fig. 2. If the Spice-subcircuits is selected, then the hierarchy is preserved and each module has its own graph structure. For a 4-bit adder example, two graphs would be constructed, one for `full_adder` (Fig. 2) and one for `4bit_adder` (Fig. 3). PLODE supports multiple levels of hierarchy, i.e. nested module instantiations.

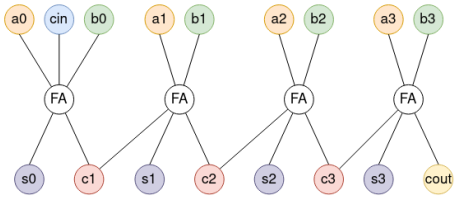

Figure 3. Graph constructed for `4bit_adder` when Spice-subcircuits option is selected

After constructing the graph structure, Spice deck is synthesized by converting each node into its Spice equivalent logic cell or subcircuit. Other than the Verilog gate primitives, the structural Verilog implementation can contain any type of logic cell that the SCL supports like compound logic gates or multiplexers.

At the end of conversion stage, the resultant Spice deck contains the Spice version of the digital circuit either in flattened or hierarchical form and ready to be prepared for the simulation.

2.3. Simulation

The simulation preparation starts with the generation of input lines. For that purpose, the user supplies the binary input vectors as a text file. There is no limitation on the number of input vectors. The time interval between two consecutive input vectors and the value of the supply voltage must also be provided to the tool. If the circuit is a sequential circuit, the clock period and the name of the clock are additionally given by the user. Using all these, PLODE inserts all inputs as piece-wise linear (PWL) voltage sources to the previously produced Spice deck.

Other than time interval and supply voltage, PLODE GUI allows the user to change any of 45 different options possible in `ngspice` like temp, reltol, abstol, gmin etc. `ngspice` is called for the resultant Spice deck to get all transistor level simulation results. At the end of the simulation, supply current vs time and output voltage vs time for each output are stored in files.

By default, `ngspice` writes all simulation results for all time instances and all circuit nodes into computer memory. For larger circuits, this quickly results in memory insufficiency. For instance, after 150ns transient simulation of c3540 circuit, ngspice was unable to further write to a 16GB memory because it has already occupied the 15GB portion of it. To avoid that, PLODE only saves the results of the output nodes to memory. This is achieved by using `save` command. This command is used to inform Spice simulator which node values will be stored to memory. As a result, both the simulation duration and the memory usage gets much better. Even for a larger circuit like c7552, memory usage didn't go above 300MB.

2.4. Analysis

The accuracy and speed of any digital circuit depends on the functional (logical) result and the circuit delay. Therefore, the analysis stage of PLODE aims at the delay detection and capturing the logical output of the circuit and it does that at the precision of Spice.

The first phase of the analysis is the detection of the signal transition times. Transition time is the exact moment when a signal transits through the half voltage, i.e. $V_{supply}/2$. It can be either high-to-low or low-to-high. In Spice extracted files, only voltage-time pairs for each signal are recorded. In order to extract the transition time, the closest voltage values to the half voltage are detected first. However, even the closest voltage values may not be exactly equal to the half voltage. At this point, the tool applies a linear interpolation to detect the exact half voltage transition time of the signal. For instance, assume that the two consecutive voltage-time values closest to the half voltage for an output node are $(v1, t1)$ and $(v2, t2)$.

$$tr\_time = (t2 - t1) \times \frac{abs(v_{half} - v1)}{abs(v2 - v1)} + t1 \qquad (1)$$

In this manner, all transition times for each input are extracted. Then, it takes the union of these to construct one array holding all possible transition times realized by the inputs. This enables the tool to be used for different input vector application scenarios. These input transition times are used in logic and delay computations. After recording all input transition times, it does the same for each output signal and stores the transition times of each output as a separate array.

For the computation of logic and delay, there is a problem that must be taken into consideration: the glitches. An output may have multiple transitions for one input transition, which is known as glitch. If not taken into consideration, glitch may result in both misleading functional test and delay results as well as wrong clock rate decisions for sequential circuits. In order to detect the stable output logic value and the correct delay, one must find the last transition at the output after a change at the input.

Fig. 4 shows the voltage-time graph for an output node of c5315 from ISCAS'85 benchmark. Input transitions occur with 10ns intervals starting from 10ns. For the second input transition at 20ns, the output first makes a small glitch but stays at high and then at about 27.5ns, it makes a high to low transition. In terms of logic value and delay result, the last transition at 27.5ns must be taken as reference. A similar situation exists at the fifth transition (after 50ns).

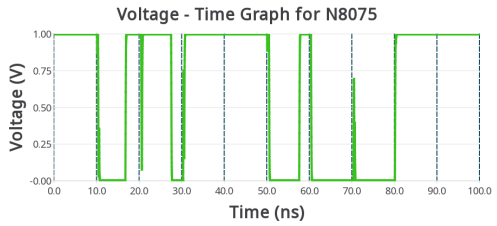

Figure 4. Voltage-time graph for N8075 output of c5315 (drawn by PLODE)

PLODE analyzes each output and detects the last output transition before the next input comes. The output value is set as the logic value after that last transition. For that particular example in Fig. 4, the tool accurately computes the logic values as $\{1, 1, 0, 1, 1, 1, 0, 0, 1, 1\}$. The delay is also computed by using that last transition and taking the difference from the input transition time. Again for N8075 of c5315, the delays plotted by the tool are shown in Fig. 5. It shows that PLODE takes into account all the glitches in Fig. 4 and computes the delay accordingly.

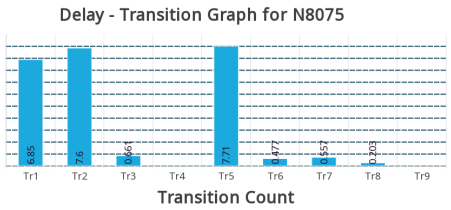

Figure 5. Delay (ns) per transition for N8075 (drawn by PLODE)

At the end of the analysis, PLODE extracts the resultant logic value of each output for each input vector. These logic values are written to a text and csv file. If the user supplies the expected outputs to PLODE as text file, it compares the actual logic values resulted from the analysis with the expected ones. It detects the number of mismatches and writes a comparison report. At that report, the actual and expected output differences with the corresponding transition numbers are listed for further analysis.

In terms of delay analysis results, a csv file containing all delays for all transitions and outputs is generated. At that csv file, the maximum delay and the corresponding output node and the transition number that is responsible from that delay are recorded too.

2.5. Single Path Mode

In most cases, the delay of a particular path, especially the critical path, is required to determine the speed of a combinational circuit or to define the clock rate for a sequential circuit. But sensitizing a specific path is a difficult task, which can be both complex and requires the gate delays a priori. For that purpose, PLODE has a path extraction module, which can build up a new circuit consisting of only the desired path and its side gates.

GUI takes the Verilog of the original circuit and the id of each interconnect on the desired path. Then, it travels in the original Verilog to get only the desired path components and remove all others. The side inputs of AND or NAND gates are made 1 and the side inputs of OR and NOR gates are made 0, so that the input pulse propagates through the desired path.

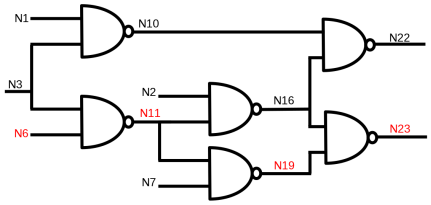

Figure 6. Circuit schematic for original c17 circuit

Fig. 6 shows the c17 circuit schematic from ISCAS'85 benchmark. Let us assume that the user wants to find the delay of path $\{N6, N11, N19, N23\}$. PLODE constructs a new Verilog file, whose schematic is as shown in Fig. 7. After this construction, it simulates the circuit by sending a pulse from $N6$, which has no choice but propagate through the desired path. The resultant delays are high-to-low and low-to-high delays for that path. It is important to note that the NAND gate in Fig. 6 with inputs $N2$ and $N11$ is not removed due to its direct effect on delay.

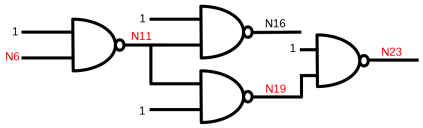

Figure 7. Schematic of the PLODE constructed Verilog

Using single path mode, any path in the circuit can be simulated for accurate path delay results without any search for input pairs that sensitizes that path. This can be very helpful for the verification of path delay based methods in different path delay based research areas like hardware security [11], statistical timing analysis [12], etc.

2.6. GUI and Visualization

Fig. 8 shows the PLODE GUI after used for the simulation and analysis of c432 circuit from ISCAS'85 bench-

mark. All desired inputs for conversion, simulation and analysis stages are given through that GUI. The stages are started by using the `Convert`, `Run Simulation` and `Analyze & Plot Logic/Delay` buttons. Using the `Compare Results` button, the number of correct and wrong outputs are presented and a text report as explained in Section 2.4 is prepared.

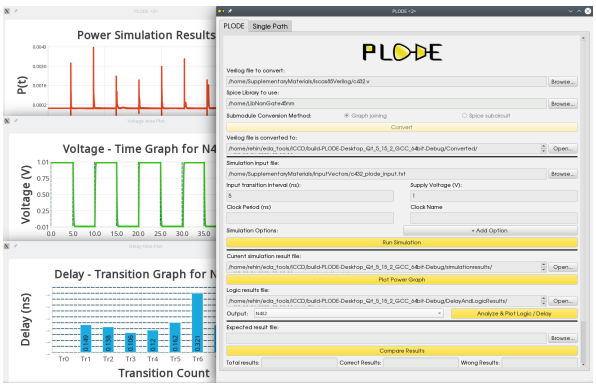

Figure 8. PLODE GUI after executed for c432

3. Experiments & Verification of PLODE

PLODE is written in C++ on `Qt`. The experiments are performed on an Intel i5-4670K 3.4GHz CPU with 16GB RAM and Opensuse Leap 15.2 operating system. In order to verify PLODE, we implement Verilog testbench files and perform logic simulations on Modelsim for verification. For each circuit, we supply the structural Verilog circuit [13] and 10 randomly generated input vectors to Modelsim. The resultant Modelsim outputs are written to text files. Then, the same Verilog files and inputs of the test circuits are given to PLODE. For the experiments, 45nm open cell library [14] and 45nm technology model [15] are used.

Table 1. PLODE experiment timings in seconds (10ns interval)

	Conversion	Simulation	Analysis	Total
c432	0.3	134.3	0.3	134.9
c499	0.6	393.6	0.5	394.7
c880	1.6	516.3	0.6	518.5
c1355	3.2	643.3	0.5	647.0
c1908	7.3	1290.3	0.4	1297.9
c2670	16.3	2267.2	2.6	2286.1
c3540	27.8	3192.5	0.5	3220.8
c5315	56.7	4912.1	2.1	4970.9
c6288	57.6	5431.8	0.5	5489.8
c7552	178.5	6660.9	2.2	6841.6

Table 1 shows the run-times for each stage of PLODE in seconds. The simulation stage includes the Spice simulation, therefore it is by far the slowest stage. Actually, if `save` command was not used the simulation duration would be much more. For instance, for c2670 the duration would become 4827 seconds instead of 2267.

PLODE analysis results for 10ns interval experiments are shown in Table 2. Using PLODE GUI, we compared the computed logic values based on Spice simulations, i.e. actual outputs, with the expected output values that are generated by Modelsim. From the total 10 input vectors, the ones that result in same output vectors with Modelsim are called *correct*,

Table 2. PLODE analysis results (10ns interval)

	Correct	Wrong	Delay	Output	Tr.
c432	10	0	1.6	N431	tr[3]
c499	10	0	0.4	N728	tr[1]
c880	10	0	2.4	N879	tr[2]
c1355	10	0	0.1	G1331	tr[4]
c1908	10	0	0.4	N2891	tr[1]
c2670	7	3	6.6	N3851	tr[1]
c3540	9	1	2.0	N5002	tr[1]
c5315	9	1	7.8	N7465	tr[2]
c6288	10	0	2.1	N6160	tr[7]
c7552	3	7	8.0	N11334	tr[5]

the output vectors different from Modelsim are called *wrong*. The difference may be in one bit or more. As the circuits get more complex, the delays increase and the 10ns input transition time interval starts to become insufficient for outputs to stabilize. Therefore, only in c2670, c3540, c5315 and c7552 the expected outputs and the actual ones do not perfectly match. According to the PLODE results, the researcher can deduce that these circuits are not suitable to drive with 10ns intervals between consecutive inputs. This table also shows other PLODE results: the max delay of the circuit in ns, which output node and which transition cause that delay respectively.

Table 3. PLODE analysis results (100ns interval)

	Correct	Wrong	Delay	Output	Tr.
c2670	10	0	65.6	N3851	tr[8]
c3540	10	0	60.1	N5360	tr[1]
c5315	10	0	67.0	N8128	tr[6]
c7552	10	0	60.5	N10908	tr[4]

As an example case, we investigate c2670 where 3 outputs were different than the expected one. By looking at PLODE comparison report, it can be seen that there is only one output that causes the error for the 3 results. This output is N3851. We repeated the PLODE experiment for all circuits resulting at least one wrong output vector. The results for the 100ns interval experiment are summarized in Table 3. As can be seen from the table, all PLODE results become identical with Modelsim outputs when 100ns interval is used instead of 10ns. This table shows that the cause for wrong outputs is the small time interval in between inputs, because each circuit in the table has maximum delay above 10ns. The maximum delay for c2670 is 65.6 and belongs to N3851, which was also the cause for the error in 10ns experiment of c2670.

In order to test PLODE on sequential circuits, s1423 and s13207 from ISCAS'89 benchmark are used. The conversion, simulation and analysis durations are shown in Table 4. For s1423 and s13207, respectively 50 and 10 input vectors are generated randomly and given in 5ns intervals. The clock period is also set as 5ns. Thus, a new input vector is given synchronously with the new clock cycle. At the end of the experiments, all PLODE outputs are compatible with Modelsim results for s1423, resulting in 100% match. On the other hand, s13207 has 152 outputs. The comparison report of PLODE

presents the output bits causing the difference for each output. Only two output bits, namely bit-24 and bit-29 among 152 bits cause the difference. From this report, we can deduce that 5ns clock cycle is not enough for output bit-24 and bit-29 to get stabilized at the expected results.

Table 4. Duration of PLODE stages in seconds

	Conversion	Simulation	Analysis	Total
s1423	5.2	790	0.1	795.2
s13207	649.8	24655	0.8	25305.6

PLODE can also extract a path including all side gates and simulate it alone to compute its delay. To test single path mode of PLODE, we assumed two paths from 32 bit carry ripple adder (`A + B = S`). The first path is the critical path of the circuit that starts with `A0` and ends at `S31`. Second path starts with `A0` again, but end at `S16`. To sensitize the paths the carry propagation must be ignited until the end of the path.

We first give the path sensitizing input vectors to full 32-bit adder circuit and PLODE computes the delay for `S31` and `S16`. Then, we compute path delays by using PLODE single path mode, where the path is extracted and simulated alone without the need for sensitizing input vectors. The resultant path delays are summarized in Table 5. The first column belongs to the full circuit simulation and the second column shows the single path mode delays of PLODE. As can be seen from the table, there is only a minimal (about 8%) difference in between.

Table 5. Path delays computed with path sensitization and single path mode of PLODE

c1908 Path	Full Circuit	Single Path	Difference
a0 - s31	2.31	2.5	%8.2
a0 - s16	1.25	1.35	%8.0

As a last experiment in order to test the hierarchy support of PLODE, we implement hierarchical Verilog design of 32-bit adder. The top module is named `adder_32bit`. It instantiates two `adder_16bit` modules, each of which instantiates four `adder_4bit` modules. Each `adder_4bit` module (Fig. 3) utilizes four `full_adder` modules (Fig. 2). The `adder_32bit` design is tested with 100 randomly generated inputs on Modelsim and the corresponding 100 output vectors are recorded for comparison. Then, this circuit is given to PLODE with Graph-joining and Spice-subcircuits options separately. For both Graph-joining and Spice-subcircuits, all resultant 100 output vectors come out to be equal to the ones that Modelsim produced, which verifies the hierarchy support of the tool.

4. Conclusion

PLODE [16] is a Spice based open source tool that can perform and visualize precise functional and timing analysis of the circuits written in structural Verilog. It can be improved in several ways. It can be accompanied by ATPG algorithms to supply input vectors sensitizing the desired paths, can be combined with a synthesis tool to be able to take behavioral Verilog as input and optimize the circuit. The statistical approaches like Monte Carlo method can be used to improve the tool further for statistical timing and power computations.

5. References

[1] Laurence W. Nagel and D.O. Pederson. Spice (simulation program with integrated circuit emphasis). Technical Report UCB/ERL M382, EECS Department, University of California, Berkeley, Apr 1973.

[2] NgSpice Circuit Simulator. `http://ngspice.sourceforge.net/`.

[3] Modelsim HDL Simulator, Last accessed: 05/2021. `https://eda.sw.siemens.com/en-US/ic/modelsim/`.

[4] Carl Burch. Logisim: A graphical system for logic circuit design and simulation. *Journal on Educational Resources in Computing (JERIC)*, 2(1):5–16, 2002.

[5] Stuart Brorson. Circuit design on your linux box using geda. *Linux Journal*, 2006(141):7, 2006.

[6] Robert K Brayton, Gary D Hachtel, Alberto Sangiovanni-Vincentelli, Fabio Somenzi, Adnan Aziz, Szu-Tsung Cheng, Stephen Edwards, Sunil Khatri, Yuji Kukimoto, Abelardo Pardo, et al. Vis: A system for verification and synthesis. In *International conference on computer aided verification*, pages 428–432. Springer, 1996.

[7] Johann Glaser Clifford Wolf and Johannes Kepler. Yosys-a free verilog synthesis suite. In *I21st Austrian Workshop on Microelectronics (Austrochip)*, 2013.

[8] Peter Jamieson, Kenneth B Kent, Farnaz Gharibian, and Lesley Shannon. Odin ii-an open-source verilog hdl synthesis tool for cad research. In *2010 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines*, pages 149–156. IEEE, 2010.

[9] F Brglez and H Fujiwara. A neutral netlist of 10 combinational benchmark circuits and a target translator into fortran. In *IEEE Int'l Symp. on Circuits and Systems (ISCAS)*, pages 659–662, 1985.

[10] Franc Brglez, David Bryan, and Krzysztof Kozminski. Combinational profiles of sequential benchmark circuits. In *IEEE International Symposium on Circuits and Systems,*, pages 1929–1934. IEEE, 1989.

[11] Yier Jin and Yiorgos Makris. Hardware trojan detection using path delay fingerprint. In *2008 IEEE International workshop on hardware-oriented security and trust*, pages 51–57. IEEE, 2008.

[12] Jing-Jia Liou, Angela Krstic, Li-C Wang, and Kwang-Ting Cheng. False-path-aware statistical timing analysis and efficient path selection for delay testing and timing validation. In *Proceedings of the 39th annual Design Automation Conference*, pages 566–569, 2002.

[13] Modelsim HDL Simulator, Last accessed: 05/2021. `http://pld.ttu.ee/~maksim/benchmarks/`.

[14] NanGate 45nm Open Cell Library. http://www.nangate.com/.

[15] Predictive Technology Model, 45nm model file. http://ptm.asu.edu/.

[16] PLODE: Precision Logic and Delay Simulator Source Code, Last accessed: 05/2021. `https://github.com/ic-cad/plode`.