Data-Parallel Simulation for Fast and Accurate Timing Validation of CMOS Circuits

Schneider, Eric; Holst, Stefan; Wen, Xiaoqing; Wunderlich, Hans-Joachim

Proceedings of the 33rd IEEE/ACM International Conference on Computer-Aided Design (ICCAD'14) San Jose, California, USA, 3-6 November 2014

url: http://dl.acm.org/citation.cfm?id=2691369

Abstract: Gate-level timing simulation of combinational CMOS circuits is the foundation of a whole array of important EDA tools such as timing analysis and power-estimation, but the demand for higher simulation accuracy drastically increases the runtime complexity of the algorithms. Data-parallel accelerators such as Graphics Processing Units (GPUs) provide vast amounts of computing performance to tackle this problem, but require careful attention to control-flow and memory access patterns. This paper proposes the novel High-Throughput Oriented Parallel Switch-level Simulator (HiTOPS), which is especially designed to take full advantage of GPUs and provides accurate time- simulation for multi-million gate designs at an unprecedented throughput. HiTOPS models timing at transistor granularity and supports all major timing-related effects found in CMOS including pattern-dependent delay, glitch filtering and transition ramps, while achieving speedups of up to two orders of magnitude compared to traditional gate-level simulators.

Preprint

Data-Parallel Simulation for Fast and Accurate Timing Validation of CMOS Circuits

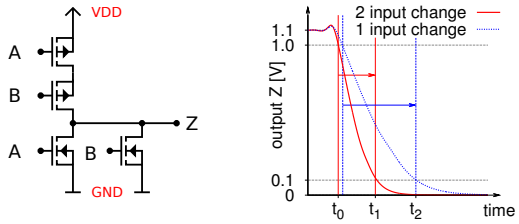
Eric Schneider*, Stefan Holst†, Xiaoqing Wen†, Hans-Joachim Wunderlich*
*University of Stuttgart, Pfaffenwaldring 47, 70569 Stuttgart, Germany
{schneiec,wu}@iti.uni-stuttgart.de
†Kyushu Institute of Technology, 680-4 Kawazu, Iizuka 820-8502, Japan
{holst,wen}@ci.kyutech.ac.jp

***Abstract*—Gate-level timing simulation of combinational CMOS circuits is the foundation of a whole array of important EDA tools such as timing analysis and power-estimation, but the demand for higher simulation accuracy drastically increases the runtime complexity of the algorithms. Data-parallel accelerators such as Graphics Processing Units (GPUs) provide vast amounts of computing performance to tackle this problem, but require careful attention to control-flow and memory access patterns. This paper proposes the novel High-Throughput Oriented Parallel Switch-level Simulator (HiTOPS), which is especially designed to take full advantage of GPUs and provides accurate time-simulation for multi-million gate designs at an unprecedented throughput. HiTOPS models timing at transistor granularity and supports all major timing-related effects found in CMOS including pattern-dependent delay, glitch filtering and transition ramps, while achieving speedups of up to two orders of magnitude compared to traditional gate-level simulators.**

I. Introduction

In Electronic Design Automation (EDA) timing-simulation of circuits is an important and widely used tool for timing validation, power estimation and delay-test assessment. With current low-power features and aggressive clocking of CMOS circuits, gate-level timing simulation has to be accurate enough to capture all important delay effects of modern CMOS cells (such as hazards or transition ramps) for useful timing, power, fault coverage and reliability estimations [1]. Take for instance the CMOS NOR-gate in Fig. 1. Simple gate-level delay models may just define rising and falling delays of this gate depending on its load. These delays may be close to the real behavior for inputs like $(0, \downarrow)$ or $(0, \uparrow)$, but with the input $(\uparrow, \uparrow)$ the gate is much faster as both NMOS-transistors discharge the load collectively. This effect has been incorporated into gate-level delay models [2, 3], so have been the cases where both input transitions do not arrive at exactly the same time or have different steepness [4]. Complex CMOS cells such as XOR-cells or multiplexers containing pass-transistor structures can exhibit unique delays for each input combination. This renders conventional analytic or look-up-table-based delay calculation approaches impractical, as they involve vast amounts of different parameters and conditions that have all to be taken into account at the same time.

Circuit timing simulation can be performed at various abstraction levels, ranging from conventional gate-level timing simulators over switch-level simulation down to the electrical level with SPICE [7]. As of today, effects like IR-drop, ground-bounce, threshold voltage variations have to be investigated with methods that heavily rely on the use of very expensive



(a) (b)

Fig. 1. a) CMOS NOR-gate. b) Output behavior with different inputs (SPICE transient analysis with a 45nm PTM [5, 6]).

low-level simulations [8]. Although it is mandatory to consider all these effects to properly predict timing, power consumption and aging under variations, it is currently way beyond the performance capabilities of these simulation algorithms to be effectively applied to large industrial-sized designs. The first hybrid GPU-accelerated SPICE simulation algorithms reached runtime improvements of up to 4x [9]. For characterizing gates even speedups of 200x have been reached on GPUs [10]. Yet, this speedup is restricted to designs with less than 22 transistors and drops significantly for larger circuits. With optimizations, simplified models and improved latency, about 10x of speedup were reported on standard CPUs [11–13]. These simulators may provide rather high level of accuracy, but they are still several orders of magnitude slower than gate-level simulation.

As a trade-off between simulation performance and accuracy various switch-level simulation approaches have been reported. Piece-wise linear models for accelerating circuit simulation have been previously investigated in many works [14–20] that are able to model current flows between individual nodes. These switch-level simulators allow the modeling of individual electrical components and therefore implicitly support all the aforementioned delay effects.

A major bottleneck for applying the above-mentioned low-level approaches to multi-million gate designs is the size of the working set per problem instance (the amount of data required during evaluation, such as frequently accessed variables). Common time simulation algorithms involve many parameters as well as vast amounts of data points during the computation, e.g. caused by signal waveform sampling, and cannot be parallelized without difficulties due to the rapidly growing memory footprint. Furthermore, since the underlying models are typically bidirectional, the mutual interplays between simulation nodes are usually expressed through differential equations that require extensive utilization of iterative solvers

for their evaluation. These calculations in turn are complex and rely on specific simulation step-widths and convergence criteria that can cause a lot of execution divergence in parallel applications, which further limit the computational throughput.

With the use of data-parallel architectures such as Graphics Processing Units (GPUs), available parallelism in circuit simulation is exploited to provide simulation speedup [21–23]. Although GPUs provide massive floating-point performance, it comes at the cost of certain control flow restrictions and a severe memory bottleneck which pose the major limitations when parallelizing simulation algorithms. On GPUs, arithmetic operations are very cheap compared to diverging control flows of data-parallel code or memory accesses, which should be kept at a minimum. Therefore, uniformity of execution and efficient memory accesses is more important than reducing the number of arithmetic operations [24].

In this work we present HiTOPS (High-Throughput Oriented Parallel Switch-Level Simulator), a novel parallel GPU-accelerated time-simulator for CMOS circuits that models timing at transistor granularity. With its sophisticated memory organization and the uniform control-flow algorithm the proposed approach is able to overcome the aforementioned bottlenecks. The remainder of this document is structured as follows: After summarizing common pitfalls and limitations of GPU architectures in Section II, we introduce the basic simulation concept of HiTOPS in Section III. Section IV presents the evaluation algorithm and the applied method of parallelization. In Section V we demonstrate that HiTOPS is able to achieve higher accuracy and throughput than conventional gate-level time-simulation algorithms and that it is capable of processing industrial-sized designs efficiently.

II. Fundamental Requirements

On data-parallel architectures, fast local memory is a very scarce resource compared to the available compute power [25]. Accesses to large global memory suffer from high latency and limited bandwidth making a careful memory management the top priority. Hence, algorithms need to have a very small working set at any point in time in order to run efficiently. The larger the working set is, the more data needs to be swapped out to global memory, which rapidly decreases performance. For instance, standard SPICE transient analysis algorithms need to access the data of a large portion of or even the complete circuit for every time step. Their working set is huge and their performance is mainly limited by memory accesses.

For optimal performance, data residing in global memory should be accessed only once and ideally in sequence to allow optimal use of pre-fetching hardware, cache memories and coalescing. Although event-based simulation algorithms have a very small working set (the time wheel), each processed event causes a signal update somewhere in a large design potentially leading to many cache misses and multiple updates of the same signal. These irregular accesses cannot be predicted or bundled rendering event-based approaches inefficient on data-parallel architectures.

In plain logic simulation, each gate in a combinational circuit is evaluated only once and in topological order to compute the result. Such a simple approach fits the requirements above quite well. The working set contains only a fraction of the whole circuit at any given time and the fixed evaluation order allows very regular memory accesses. It has been shown that this principle can be applied to timing simulation to reach enormous throughput on GPUs [23]. The key to apply the same principle to switch-level simulation is to use a circuit model that allows topological ordering of its basic components. The input values to each component must be independent of the state of the component itself, and the output values of a component must be independent of the states of all succeeding components. These requirements are not fulfilled by existing switch-level or electrical circuit models, because the output voltage of a CMOS cell depends on the wire resistance, the capacitive load and even the charging state of the connected circuitry.

III. Simulation Model for CMOS Circuits

HiTOPS considers individual transistors within the CMOS cells and uses continuously valued voltage waveforms that closely match the actual voltages at the gates of the transistors. The switching times of individual transistors are determined by the intersection points of these voltage waveforms with their threshold voltages.

A. Channel-Connected Components

The goal is to partition the transistor netlist into components that can be simulated in topological order. In CMOS circuits, so-called channel-connected components [26] have exactly this property. A channel-connected component is a sub-network in a transistor netlist in which current can flow freely between nodes. Fig. 2 shows a transistor netlist with a highlighted channel-connected component. Ideal transistors do not allow any current flow from the gate to its other terminals and all connections to an ideal power supply network draw currents independently from each other. Therefore, channel-connected components are found by starting with an arbitrary net and traverse from it until a transistor-gate or the power supply network is reached.

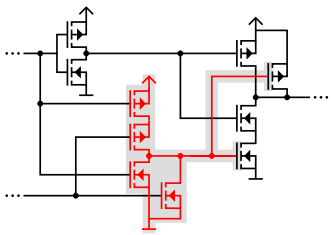


Fig. 2. A channel-connected component in a transistor netlist.

The state of the channel is controlled by the gate voltages of its associated transistors and it determines the gate voltages for succeeding channel-connected components. If the transistor netlist itself does not contain any loops, the channel-connected components can be evaluated in topological order to calculate all gate voltages in the netlist in a single pass. The behavior is governed by numerous intended and parasitic effects which can be approximated with various degrees of accuracy [14, 19]. But despite all these effects, the transitions of the gate voltages can still be well approximated by simple exponential functions. This is shown by a quick SPICE transient analysis of some CMOS inverters and a fitting to ideal exponential curves in

Fig. 3. Hence, for the purpose of accurate timing simulation, such curves contain sufficient information to cover the most important delay effects in CMOS circuits.

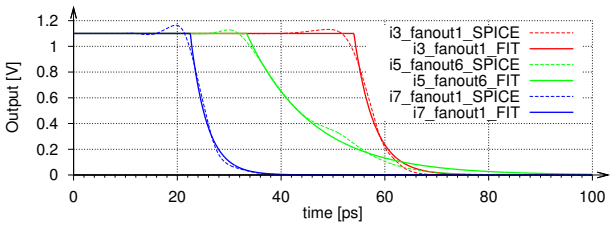


Fig. 3. SPICE transient analysis (45nm PTM [5, 6]) of CMOS inverters under different loads and fitting of exponential functions to the SPICE result.

Arbitrary complex channel-connected components such as 6-transistor XOR-Cells or OR-AND-Inverters (OAI) with different driving strengths are characterized using SPICE simulations to find the relationship between input and output. The parameters for the exponential curves that fit the voltage transitions best in each situation are then input to HiTOPS for fast simulation. For easier presentation of the core principles of HiTOPS, we will introduce an intuitive model for handling CMOS gates in the next section. The same ideas are applicable to complex gates and more powerful models of general channel-connected components as well.

B. Resistor-Resistor-Capacitor Cells

HiTOPS uses Resistor-Resistor-Capacitor (RRC-) cells as shown in Fig. 4 for computation, which is a simple and intuitive unidirectional model for describing the behavior of channel-connected components in CMOS gates in a compact way. CMOS gates contain pull-up and pull-down networks, that connect to VDD as well as GND, and which are represented by a resistive voltage divider (R_u, R_d) charging or discharging a lumped capacity C_{load} over time.

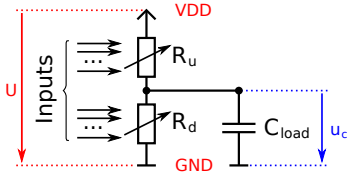


Fig. 4. The RRC-cell.

If the values of both R_u and R_d are constant between time t_p and t_{p+1} (with $p \geq 0$), the output voltage $u_c(t)$ at the output capacitor will follow an exponential curve starting from an initial voltage $u_c(t_p)$ towards the stationary voltage $\overline{u} = S \cdot U + \text{GND}$, where $S = \frac{R_d}{R_u + R_d}$ is the ratio of the voltage divider and $U = \text{VDD} - \text{GND}$. With the time constant $\tau = SR_uC$, the exponential curve within $t_p \leq t \leq t_{p+1}$ can be computed by:

$$u_c(t) = (u_c(t_p) - \overline{u})e^{-\frac{t-t_p}{\tau}} + \overline{u}. \tag{1}$$

To calculate the values of R_u and R_d, each transistor present in a gate is replaced by a voltage-controlled resistor that is described by a 3-tuple $T = (U_{th}, R^0, R^1)$. Each transistor is considered a perfect switch, that changes its source-drain resistance R_T between the values R^0 and R^1 depending on its threshold voltage U_{th} and the current gate voltage u_T:

$$R_T(u_T) = \begin{cases} R^0 & u_T < U_{th} \\ R^1 & u_T \geq U_{th} \end{cases}. \tag{2}$$

Here, U_{th} is defined as the gate potential over ground GND at which the transistor changes its state. This way, pull-up and pull-down transistors can be treated in very much the same way. NMOS-transistors have $R^0 > R^1$, and PMOS-transistors have $R^1 > R^0$. High resistance values indicate a blocking transistor, while a low resistance implies a conducting state. A wire resistance $R_W \geq 0$ also contributes to R_u and R_d, which are then computed during runtime based on all current transistor resistances $R_T(u_T)$ by applying Kirchhoff's laws for parallel and series circuits to the pull-up and pull-down meshes respectively.

In the case of the NOR-gate from Fig. 5, the final resistances are $R_u = R'_u + R_W \frac{R'_u + R'_d}{R'_d}$ and $R_d = R'_d + R_W \frac{R'_u + R'_d}{R'_u}$ with $R'_u = R^P_A + R^P_B$ and $R'_d = \frac{R^N_A R^N_B}{R^N_A + R^N_B}$.

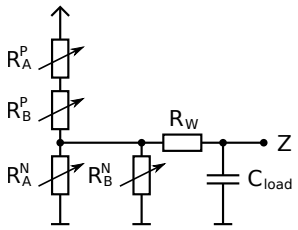


Fig. 5. Equivalent circuit model of the NOR-gate.

Since between two consecutive transistor switches the resistances are assumed to be constant, the voltage level at C_{load} will follow an exponential curve with the parameters derived before. It is easy to see, how this model can keep continuous track of the charge at C_{load} over time and how multiple input transitions in close succession can result in glitch filtering or faster switches as depicted in Fig. 1. Note that Eq. (1) supports customized VDD and GND levels for individual RRC-Cells and hence allows to consider IR-drop, ground-bounce and threshold voltage variations as well.

C. Waveform Representation

In order to describe signal changes or switching histories, HiTOPS utilizes an efficient concept similar to [23], that has been extended by piecewise waveform approximation with curve segments. In an RRC-cell, each transistor switch causes a new charging curve to appear at the output. The start of a new curve is called *pivot*. A pivot p and its associated curve are entirely described by three parameters:

- t_p: The time of the pivoting point. This time marks the end of the previous exponential curve $(p-1)$ and the start of the new one (p).
- $\overline{u}_p$: The stationary voltage the new curve p approaches, calculated from the new resistances of the voltage divider, the supply voltage U and GND.
- τ_p: The time constant of the new curve p, calculated from the resistances of the voltage divider and the lumped load capacity.

A waveform is a list of pivot points $(t, \overline{u}, \tau)$ ordered temporally from the earliest to the latest. Before the first ordinary pivot, the waveform has a constant initial voltage encoded as $(-, u_{init}, -)$ at the head of the sorted list. The list is terminated by $(\infty, -, -)$ and the waveform approaches the voltage $\overline{u}$ of the last ordinary pivot. Fig. 6 shows plots and encodings of three example waveforms. The pivot representation completely avoids sampling and allows to model continuous-valued voltages efficiently with compact storage requirements similar to traditional timing simulation.

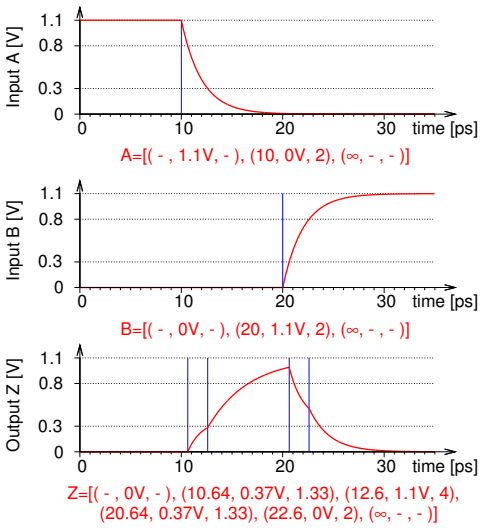


Fig. 6. Example waveforms and their encoding for inputs and output of a two-input NOR-gate. Ordinary pivots are marked by (blue) vertical lines.

IV. HiTOPS Simulation Algorithm

A. RRC-Cell Evaluation

The following algorithm takes all waveforms controlling a single RRC-cell, calculates all transistor switch times by intersecting the waveforms with the appropriate threshold voltages, and generates a new waveform describing the output voltages over time of the cell. A single input transition may generate multiple transistor switches at distinct times. All switches have to be managed and sorted from oldest to newest to generate the correct output waveform, which are clearly the most challenging tasks of the complete evaluation procedure.

An efficient solution is given in Algorithm 1. Since the pivot points in the input waveforms are already sorted, the evaluation follows a merge-sort approach which requires to read and write waveforms only once in sequential order and allows to keep the local working set as small as possible. The evaluation progress is controlled by two time points $[t_{min}, t_{max}]$ defining a time window that slides across all input waveforms from the beginning to the end. The window start time t_{min} is increased with every transistor switching event processed and t_{max} is increased with every new input pivot loaded from memory. Local memory contains two tables, the *pivot table* and the *transistor table*, to track the cell's state within $[t_{min}, t_{max}]$. The pivot table always contains all necessary data to describe the input voltages within the current window, and the transistor table always contains the state of all transistors at t_{min} and all switching times until t_{max}.

Algorithm 1: RRC-cell evaluation algorithm

1 **foreach** *row in pivot table* **do**
2 Load first t_p, τ_p, $\overline{u}_p$ and t_{p+1} from memory
3 $u_p := \overline{u}_p$ and $u_{p+1} := \overline{u}_p$
4 **end**
5 $t_{min} := -\infty$ and $t_{max} := \min\{t_{p+1}$ in pivot table$\}$
6 **foreach** *column k in transistor table* **do**
7 $R_T^k := R_T(u_p$ from pivot table), using Eq. (2)
8 $t_{next}^k := \infty$
9 **end**
10 Output $(-, SU + \text{GND}, -)$, with S as the voltage divider ratio
11 **repeat**
12 $t_{min} := t_{max}$
13 **foreach** *row in pivot table with* $t_{p+1} = t_{max}$ **do**
14 Set $u_p := u_{p+1}$ and $t_p := t_{p+1}$
15 Load next τ_p, $\overline{u}_p$ and t_{p+1} from memory
16 Calculate u_{p+1} using Eq. (1)
17 **end**
18 $t_{max} := \min\{t_{p+1}$ in pivot table$\}$
19 **foreach** *column k in transistor table* **do**
20 $t := t_p - \tau_p \log\left(\frac{U_{th}^k - \overline{u}_p}{u_p - \overline{u}_p}\right)$ with data from pivot table
21 **if** $t_{min} < t \leq t_{max}$ **then**
22 $t_{next}^k := t$
23 **else**
24 $t_{next}^k := \infty$
25 **end**
26 **end**
27 **foreach** *column k in transistor table with* $t_{next}^k < \infty$ *in ascending order of* t_{next} **do**
28 $R_T^k := R_T(u_{p+1}$ from pivot table), using Eq. (2)
29 Output $(t_{next}^k, SU + \text{GND}, SR_uC)$
30 $t_{min} := t_{next}^k$
31 **end**
32 **until** $t_{max} = \infty$
33 Output $(\infty, -, -)$

We now will demonstrate the operation of this algorithm step-by-step on a two-input NOR-gate (Table I). For the sake of simplicity, open transistors have a resistance of $20M\Omega$ and conducting ones have $2k\Omega$ in this example. The threshold voltages for the two PMOS (NMOS) transistors are $0.8V$ ($0.3V$) over ground. The lumped load capacitance is $C_{load} = 1pF$ and the supply voltage is $U = 1.1V$ (VDD $= 1.1V$, GND $= 0V$). The stimuli for both inputs are the same as depicted in Fig. 6.

Table I denotes the performed operations, the changes in the pivot and transistor table and the result of each step. The pivot table contains two rows, one for each input of the NOR-gate. Each row contains the voltage at the pivot point u_p, the time of the pivot t_p, its stationary voltage $\overline{u}_p$ and time constant τ_p, and the voltage and time of the next pivot u_{p+1}, t_{p+1}. The transistor table contains one column for each transistor (four in this case) and two rows. The row R_T holds the resistance at time t_{min} for each transistor, and row t_{next} holds the time of the next switch within $[t_{min}, t_{max}]$ or ∞ if there is no switch.

Step 1: Initialization. (Algorithm 1, lines 1–10) The first pivots from input waveforms A and B are loaded into t_p, $\overline{u}_p$, and τ_p of the pivot table. These pivots only define the initial stable voltages, so u_p and u_{p+1} are set to the same value as $\overline{u}_p$. The times of the next pivots in each waveform are loaded into t_{p+1}, and t_{max} is set to the minimum of these times. The R_T

TABLE I. EXECUTION OF ALGORITHM 1 FOR A NOR-GATE. OPERATIONS AND CONTENTS OF PIVOT AND TRANSISTOR TABLE AFTER EACH STEP. THE INPUT AND OUTPUT WAVEFORMS ARE THE SAME AS SHOWN IN FIG. 6.

Step		Pivot table						t_{max}	Transistor table					Action
			u_p	$(t_p,$	$\overline{u}_p,$	$\tau_p)$	u_{p+1}	t_{p+1}			T_A^P	T_B^P	T_A^N	T_B^N	
1	load initial pivot (0), init R_T, $t_{min} = -\infty$.	A:	1.1	–	1.1	–	1.1	10	10	R_T:	20M	2k	2k	20M	output (–,0,–)
		B:	0	–	0	–	0	20		t_{next}:	∞	∞	∞	∞	
2	advance A ($p = 1$), $t_{min} = 10$, calc t_{next}.	A:	1.1	10	0	2	0	∞	20	R_T:	20M	2k	2k	20M	
		B:	0	–	0	–	0	20		t_{next}:	12.60	∞	10.64	∞	
3	process T_A^N switch, generate pivot.									R_T:	20M	2k	20M	20M	output (10.64, 0.37, 1.33)
										t_{next}:	12.60	∞	∞	∞	set $t_{min} = 10.64$
4	process T_A^P switch, generate pivot.									R_T:	2k	2k	20M	20M	output (12.60, 1.1, 4.00)
										t_{next}:	∞	∞	∞	∞	set $t_{min} = 12.60$
5	advance B ($p = 1$), $t_{min} = 20$, calc t_{next}.	A:	1.1	10	0	2	0	∞	∞	R_T:	2k	2k	20M	20M	
		B:	0	20	1.1	2	1.1	∞		t_{next}:	∞	20.64	∞	22.60	
6	process T_B^P switch, generate pivot.									R_T:	2k	20M	20M	20M	output (20.64, 0.37, 1.33)
										t_{next}:	∞	∞	∞	22.60	set $t_{min} = 20.64$
7	process T_B^N switch, generate pivot.									R_T:	2k	20M	20M	2k	output (22.60, 0, 2.00)
										t_{next}:	∞	∞	∞	∞	set $t_{min} = 22.60$
8	$t_{max} = \infty$ $\rightarrow$ terminate.									R_T:	2k	20M	20M	2k	output (∞,–,–)
										t_{next}:	∞	∞	∞	∞	

row of the transistor table is filled with the resistance values corresponding to the initial transistor states. All voltages are stable until t_{max} and no switches will take place, so the t_{next} row is initialized to ∞. The voltage divider ratio is computed from the transistor resistances and the resulting stable output voltage is written to the output waveform.

Step 2a: Advance window. (lines 12–18) Everything until time t_{max} has been processed. t_{min} is set to t_{max} and the earliest next pivot is loaded into $t_p, \overline{u}_p, \tau_p, t_{p+1}$ of the pivot table (row A in this case). u_p is set to u_{p+1} and the new ending voltage u_{p+1} is calculated from available parameters with Eq. (1). t_{max} is again the minimum t_{p+1}, which is 20 in our example.

Step 2b: Compute transistor switch times. (lines 19–26) Each t_{next}^k in the transistor table is set to the time where a curve from pivot table crosses the threshold voltage of a transistor k. Whenever there is no switch of a transistor k within the window $[t_{min}, t_{max}]$, t_{next}^k is ∞. For a transistor, only a single switch per pivot is possible, since the monotonously increasing or decreasing curve segments in the pivot table can cross the threshold level of a transistor at most once.

Steps 3 and 4: Generate output pivots in order. (lines 27–31) The earliest transistor switch will be executed, updating its resistance R_T^k. The respective values for $\overline{u}$ and τ are then calculated with the new resistances of the voltage divider and the new pivot $(t_{next}, \overline{u}, \tau)$ is added to the output waveform. After this, t_{min} advances to t_{next}. In the example, first the switch of T_A^N at 10.64 is consumed, then the switch of T_A^P at 12.60.

Step 5: Advance and compute switch times. Every event in the window has been processed and the pivot table is updated like in Step 2a (this time row B). The loaded segments cross the thresholds of T_B^P and T_B^N within the time window $[20, \infty]$, and in **steps 6 and 7**, the corresponding pivots are added to the output waveform like in steps 3 and 4.

Step 8: Termination condition. (lines 32 and 33) t_{max} has reached ∞, which indicates that all input pivots have been processed. The procedure terminates by adding the delimiter symbol $(\infty, -, -)$ to the output waveform.

Fig. 6 shows the plot of the resulting output waveform with detailed timing and voltage information. During the simulation of the gate, a hazard has been generated, that might get filtered later on depending on the parameters of succeeding RRC-cells. It is also easy to see, how the algorithm supports the behavior in the introductory example (Fig. 1). Overall, this algorithm supports very complex timing behavior with very simple control flow and minimal local memory requirements.

B. Data-Parallel Simulation

HiTOPS follows the same approach as in [23] in order to execute Algorithm 1 on GPUs for many waveforms and RRC-cells at the same time in a data-parallel fashion (see Fig. 7).

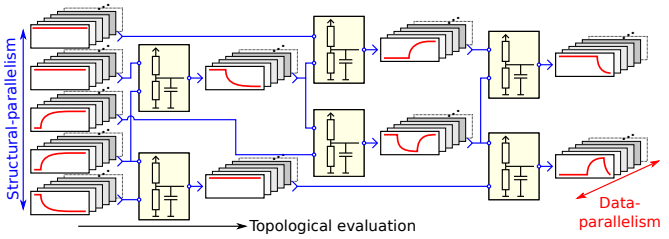


Fig. 7. HiTOPS simulation.

As a pre-processing step, all RRC-cells are sorted in topological order. Cells residing on the same topological level are then evaluated in parallel by different threads, because they are mutually data-independent. The many independent input waveforms applied to the circuit allow to exploit a second dimension of parallelism. During execution, two-dimensional

grids of thread-blocks are spawned for each topological level in-order, with each thread in the grid computing the output waveform of a particular cell and given stimulus. Before the actual HiTOPS simulation, a fast RT-level simulator creates traces for all flip-flops and inputs of the design. With all transitions of state elements known, they are split into individual and independent waveforms for each clock cycle. HiTOPS then propagates all these waveforms through the combinational circuit in parallel to report timing violations, power consumption or other information depending on the application.

During the parallel evaluation of a cell the different threads of an execution batch each operate on separate stimuli waveforms. With the lock-step execution of the thread scheduler, many of the active threads process their inputs in the same order and also access the same pivot index in the respective input waveforms simultaneously. This fact is exploited in the organization of the waveform memory in order to optimize memory accesses. Fig. 8 shows the alignment of the waveform memory for a single cell with waveforms being stored vertically. This way, the threads in the batch access pivot elements with consecutive memory addresses. These accesses are coalesced by the scheduler and the data is cached, hence resulting in a minimum amount of memory transactions for the input processing.

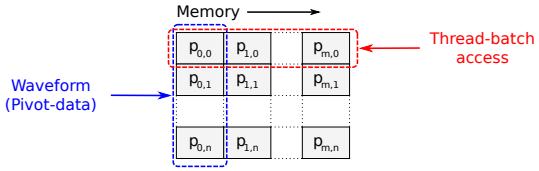


Fig. 8. Memory organization for coalesced pivot-data access.

In direct comparison, the evaluation of a single clock cycle might be slower than latency-optimized algorithms running on CPUs. Yet, HiTOPS can handle many thousands of independent assignments at the same time to reach enormous throughput on GPU architectures. With the small memory footprint of both RRC-cell and efficient waveform descriptions, HiTOPS is able to fit multi-million gate circuits efficiently on a GPU device. The simplicity and uniformity of the evaluation flow as well as a sophisticated memory organization allow to coalesce and reduce memory accesses and operations to a minimum in order to achieve maximum simulation speedup.

V. Experimental Results

HiTOPS was implemented and executed on a NVIDIA® CUDA™Kepler series GPU with 5GB of global memory and a clock of 700MHz. The host system contained Intel® Xeon® processors with 2.8GHz clock rate and 256GB RAM.

Since HiTOPS is designed to handle large, industrial designs in a high throughput fashion, our benchmark set therefore includes only the largest ITC'99 circuits and industrial designs provided by NXP. Conventional simulation approaches with their accelerated variants and switch-level simulators are not designed to work on such large designs. In order to get an impression of the runtime performance, we compare the simulation time of HiTOPS to a commercial event-based gate-level time-simulation tool with a basic unit-delay model applied (cf. Table II).

TABLE II. Runtime comparison for 10,000 patterns.

Circuit	Gates	Cells	Gate-level	Cold-Run		Re-Run	
				t_{init}	X	t_{full}	X
b17	39k	48k	0:17h	38.49s	25x	15.34s	63x
b18	132k	157k	1:27h	2:56m	29x	59.99s	86x
b19	265k	317k	3:12h	6:31m	29x	1:57m	98x
p45k	49k	77k	0:29h	53.68s	31x	23.86s	71x
p77k	79k	127k	3:36h	3:49m	56x	1:19m	163x
p78k	81k	128k	2:05h	2:59m	41x	1:08m	109x
p81k	117k	192k	1:12h	2:08m	33x	48.92s	87x
p89k	98k	157k	0:58h	1:48m	32x	43.31s	79x
p100k	108k	172k	1:26h	2:35m	33x	57.31s	89x
p141k	194k	307k	2:31h	5:09m	29x	1:40m	91x
p239k	296k	477k	5:24h	0:14h	24x	3:02m	106x
p267k	305k	452k	3:13h	7:27m	25x	2:07m	91x
p279k	324k	502k	4:03h	0:11h	22x	2:19m	104x
p295k	328k	536k	3:21h	9:43m	20x	2:32m	79x
p330k	391k	620k	6:35h	0:15h	27x	3:18m	120x
p378k	404k	639k	17:33h	0:20h	54x	5:36m	188x
p388k	538k	874k	11:16h	0:31h	22x	5:52m	115x
p418k	499k	768k	7:37h	0:25h	18x	3:51m	118x
p469k	77k	113k	31:53h	7:45m	247x	2:50m	675x
p483k	582k	891k	17:25h	0:30h	35x	5:55m	176x
p500k	557k	902k	13:31h	0:33h	24x	5:30m	147x
p533k	729k	1.2M	18:21h	0:39h	28x	7:36m	144x
p874k	802k	1.2M	13:46h	1:01h	13x	6:13m	133x
p951k	1.2M	1.9M	33:45h	1:32h	22x	0:11h	195x
p1522k	1.2M	2.0M	37:19h	1:40h	22x	0:13h	184x

In a pre-processing step, the benchmarks were mapped to the NanGate 45nm Open Cell Library [27]. Their state-elements were removed, thus leaving only the combinational logic for simulation. The combinational logic was partitioned into channel-connected components and each component type was characterized by SPICE simulations to determine appropriate values for RRC-cell parameters. As shown in column two and three, the number of resulting RRC-cells is roughly twice the number of gates in the circuit.

For the evaluation of the speedup, we applied 10,000 randomly generated input stimuli in succession to each benchmark circuit and compared the runtime results of HiTOPS with the event-based gate-level time simulation. For HiTOPS, two runtimes are reported in Table II, because just as the approach in [23], the waveform allocation in the global memory is optimized during runtime based on the number of observed pivots. In the experiments, an initial waveform capacity of 16 pivots was chosen. Without previous knowledge of the expected number of pivots per signal, HiTOPS needs to spend some time on memory management [23] and consequently does not reach maximum performance.

In the worst case (column *Cold-Run*) speedups of 13–247x are achieved. In the second run (column *Re-Run*), all waveforms fit into perfectly into previously allocated memories, which allows the simulator to run with no calibration overhead, increasing speedups to 63–675x. The circuit p469k shows unusually high speedups, because its high number of hazards affects the performance of event-based simulation much more than in HiTOPS. By striving for maximum throughput, HiTOPS is able to outperform basic gate-level simulators by far, both in terms of runtime and in terms of accuracy of the delay model.

VI. Conclusions

The demand for higher simulation accuracy drastically increases the runtime complexity of conventional timing-simulation algorithms. For the first time, this paper proposes the novel High-Throughput Oriented Parallel Switch-level Simulator (HiTOPS), which enables fast and accurate timing-simulation for multi-million gate designs. HiTOPS supports all major timing-related effects found in CMOS including pattern-dependent delay, glitch filtering as well as transition ramps, and it is especially designed to take advantage of the vast amounts of computing performance provided by data-parallel accelerators such as Graphics Processing Units (GPUs). By exploiting structural- and data-parallelism, careful memory access patterns and strict uniformity in the code execution, the proposed simulator is able to achieve unprecedented throughput and outperforms traditional gate-level simulators by far in terms of both accuracy and performance. With its detailed delay model, HiTOPS even enters the domain of switch-level simulation and enables numerous design analyses and characterization applications for industrial-sized designs that are currently infeasible with conventional simulators.

VII. Acknowledgments

This work has been funded by the German Research Foundation (DFG) under contract WU 245/16-1.

References

[1] The International Technology Roadmap for Semiconductors: 2013. http://www.itrs.net/Links/2013ITRS/Home2013.htm, 2014.

[2] E. Melcher, W. Röthig, and M. Dana. Multiple input transitions in CMOS gates. *Microprocessing and Microprogramming*, 35(1–5):683–690, 1992.

[3] L.-C. Chen, S.K. Gupta, and M.A. Breuer. A new gate delay model for simultaneous switching and its applications. In *Proc. 38th Design Automation Conf. (DAC)*, pp. 289–294, 2001.

[4] M. J. Bellido, J. Juan, and M. Valencia. *Logic-timing Simulation and the Degradation Delay Model*. Imperial College Press, 2006.

[5] Predictive Technology Model (PTM). http://ptm.asu.edu, 2014.

[6] W. Zhao and Y. Cao. New Generation of Predictive Technology Model for Sub-45 nm Early Design Exploration. *IEEE Trans. on Electron Devices*, 53(11):2816–2823, Nov. 2006.

[7] L. 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.

[8] J. Jiang, M. Aparicio, M. Comte, F. Azais, M. Renovell, and I. Polian. MIRID: Mixed-Mode IR-Drop Induced Delay Simulator. In *Proc. IEEE 22nd Asian Test Symp. (ATS)*, pp. 155–160, 2013.

[9] K. Gulati, J.F. Croix, S.P. Khatri, and R. Shastry. Fast circuit simulation on graphics processing units. In *Proc. 14th Asia and South Pacific Design Automation Conf. (ASP-DAC)*, pp. 403–408, 2009.

[10] L. Han, X. Zhao, and Z. Feng. TinySPICE: A parallel SPICE simulator on GPU for massively repeated small circuit simulations. In *Proc. ACM/EDAC/IEEE 50th Design Automation Conf. (DAC)*, pp. 1–8, 2013.

[11] Z. Li and C.-J.R. Shi. SILCA: SPICE-accurate iterative linear-centric analysis for efficient time-domain simulation of VLSI circuits with strong parasitic couplings. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 25(6):1087–1103, Jun. 2006.

[12] E. Acar, F. Dartu, and L.T. Pileggi. TETA: transistor-level waveform evaluation for timing analysis. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 21(5):605–616, May 2002.

[13] Z. Zhu, K. Rouz, M. Borah, C.-K. Cheng, and E. S. Kuh. Efficient transient simulation for transistor-level analysis. In *Proc. Asia and South Pacific Design Automation Conf. (ASP-DAC)*, pp. 240–243, 2005.

[14] C. Visweswariah and R.A. Rohrer. Piecewise approximate circuit simulation. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 10(7):861–870, Jul. 1991.

[15] B. Chawla, H. Gummel, and P. Kozak. MOTIS-An MOS timing simulator. *IEEE Trans. on Circuits and Systems*, 22(12):901–910, Dec. 1975.

[16] L.M. Brocco, S.P. McCormick, and J. Allen. Macromodeling CMOS circuits for timing simulation. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 7(12):1237–1249, Dec. 1988.

[17] S. Lin, E.S. Kuh, and M. Marek-Sadowska. Stepwise equivalent conductance circuit simulation technique. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 12(5):672–683, May 1993.

[18] J. Qian, S. Pullela, and L. Pillage. Modeling the Effective capacitance for the RC interconnect of CMOS gates. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 13(12):1526–1535, Dec. 1994.

[19] R. Kao. *Piecewise Linear Models for Switch-level Simulation*. WRL Research Report 92/5. Western Research Laboratory, Departments of Electrical Engineering and Computer Science, Stanford University, 1992.

[20] A. Devgan and R.A. Rohrer. Adaptively controlled explicit simulation. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 13(6):746–762, Jun. 1994.

[21] K. Gulati and S.P. Khatri. Towards acceleration of fault simulation using Graphics Processing Units. In *Proc. ACM/IEEE 45th Design Automation Conf. (DAC)*, pp. 822–827, 2008.

[22] M.A. Kochte, M. Schaal, H. Wunderlich, and C.G. Zoellin. Efficient fault simulation on many-core processors. In *Proc. ACM/IEEE 47th Design Automation Conf. (DAC)*, pp. 380–385, 2010.

[23] S. Holst, E. Schneider, and H. Wunderlich. Scan Test Power Simulation on GPGPUs. In *Proc. IEEE 21st Asian Test Symp. (ATS)*, pp. 155–160, 2012.

[24] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips. GPU Computing. *Proceedings of the IEEE*, 96(5):879–899, 2008.

[25] High Performance Computing (HPC) and Supercomputing — NVIDIA Tesla — NVIDIA. http://www.nvidia.com/object/tesla-supercomputing-solutions.html, 2014.

[26] P. Debefve, F. Odeh, and A. E. Ruehli. Waveform Techniques. In A. E. Ruehli, ed., *Circuit Analysis, Simulation and Design*, vol. 3 of *Advances in CAD for VLSI*, ch. 8, pp. 41–127. Elsevier Science B.V., 1987.

[27] NanGate 45nm Open Cell Library. http://www.nangate.com, 2014.