Hypervised Transient SPICE Simulations of Large Netlists & Workloads on Multi-Processor Systems

Grigorios Lyras, Dimitrios Rodopoulos, Antonis Papanikolaou and Dimitrios Soudris
National Technical University of Athens – School of Electrical and Computer Engineering
MICROprocessors and Digital Systems LABoratory
9, Heroon Polytechneiou str., Zographou Campus, 157 80 Athens, Greece
Conctact Email: drodo@microlab.ntua.gr

***Abstract*—The need for detailed simulation of integrated circuits has received significant attention since the early stages of design automation. Given the increasing device integration, these simulations have extreme memory footprints, especially within unified memory hierarchies. This paper overcomes the infeasible memory demands of modern circuit simulators. Structural partitioning of the netlist and temporal partitioning of the input signals allow distributed execution with minimal memory requirements. The proposed framework is validated with simulations of a circuit with more than 10^6 MOSFET devices. In comparison to a commercial tool, we observe minimal error and even $\times 2.35$ speedup for moderate netlist sizes. The proposed framework is proven highly reusable across a variety of execution platforms.**

I. Introduction

Computer-aided simulation of circuit activity is a key issue in the verification of a wide range of electric and electronic systems [1]. Especially in the case of integrated circuits (ICs), a definitive piece of work is the Simulation Program with Integrated Circuit Emphasis (SPICE) [2]. This software has long been considered as the standard for detailed IC simulation. SPICE has also triggered a wide variety of research. A significant portion of that research aims at the acceleration of SPICE simulations, as a result of *increasing device inventories*.

Device variability also creates demand for computationally viable SPICE simulations. Deep sub-micron devices are reported to exhibit variability during system lifetime, with both a stochastic and a workload-dependent component [3]. Transient simulations of representative workloads are required to account for time- and workload-dependent variability [4]. Hence, *the memory requirements of transient SPICE simulations are increasing with device inventories and extended workloads*.

State-of-the-art (SotA) SPICE optimizations exhibit a clear trend towards parallel execution across various units of execution (UEs). This is not surprising, if we consider the increasing number of cores found in modern large-scale processing systems [5]. The concepts of *node* or *branch tearing* are very appealing when parallelizing SPICE. They involve the partitioning of the initial netlist to smaller subcircuits. Each subcircuit can be solved independently and contribute to the solution of the initial netlist. Many papers look into optimizing node tearing or parallelize intensive execution stages. Other papers, deal with parallel mapping of SPICE on specific hardware. However, in all these cases, the demand for main memory is made concurrently to the executing platform, which may prove incapable of fulfilling this requirement.

In this paper, we differentiate from the state-of-the-art by enabling massive SPICE simulations that overcome memory capacity issues. Simulation threads with minimal memory requirements are distributed across available UEs. We propose the temporal partitioning of the netlist's primary input signals, in what will be referred as *workload tearing*. When combined with node tearing, this concept enables the execution of small and independent SPICE instances across the available UEs. These instances take maximal advantage of the infrastructure capabilities with minimal communication overhead. A hypervisor has been designed for the dispatching of these SPICE instances and the reconciliation of simulation results. The proposed framework imposes minimum hardware constraints on the executing platform and is highly reusable. It has been seamlessly integrated in an experimental cloud chip, a set of virtual machines and a regular multicore server. Its accuracy and performance are evaluated against the commercial tool HSPICE, which supports multi-threaded execution.

In the next Section, we summarize the SotA on SPICE optimizations. In Section III we elaborate on the proposed structural and temporal partitioning. We also present the hypervisor, which maintains high level control of our framework. Section IV presents the inspected multi-core platforms, the netlist benchmark that was used and simulation results. Finally, conclusions are summarized in Section V.

II. Related Work

Optimizations of SPICE performance can be split into two major categories. Some optimizations target SPICE execution on a single UE. Other solutions deal with the distribution of a SPICE execution across various UEs. Naturally, the former category attracted significant amount of research, even from the initial development stages of the SPICE software [6]. The ideas of Node [7] and Branch Tearing [8] were introduced to physically partition the initial simulation problem to independent subproblems. Decomposition based on the unknown variables of the problem's differential and algebraic equations was introduced in the Waveform Relaxation method [9].

As the multi-core trend materialized, research focused on SPICE parallelization. The data-level parallelism of tearing

SCC infrastructure provided by Intel Labs Braunschweig, Germany in the context of the EU FP7-INFSO-IST-248789 TRAMS Project.
978-3-9815370-0-0/DATE13/©2013 EDAA

or decomposition methods has been heavily exploited. The authors of [10] are mimicking a transmission line, to emulate the signal propagation delay between nodes of different circuit partitions. An alternative to node tearing is also proposed in [11], aiming at a reduced number of connections between partitions. Finally, [12] proposes alternative partitioning criteria. The design automation industry is following the trend of mulitcore SPICE simulations providing a range of related products [13]. Many SPICE acceleration attempts are using specific hardware to exploit inherent data-level parallelism. Field Programmable Gate Arrays (FPGAs) have been used to parallelize the tasks performed by the SPICE simulator [14]. Graphics Processing Units (GPUs) have also been used for the acceleration of transistor model evaluation [15].

In view of the related work on SPICE acceleration, it is evident that netlist partitioning has received significant attention. Even though optimized, the partitioning of the circuit is not enough to avoid the violation of the memory constraints imposed by the executing hardware [11]. Other approaches that propose the use of customized hardware to perform the simulations (e.g. GPUs or FPGAs) reduce the versatility of the parallel SPICE implementations. Our parallel framework reduces the memory footprint of the simulation on each UE, while remaining highly reusable across processing systems usually found in academic or industry environments.

III. Hypervised SPICE Simulations

A. Simulation Framework

Assume a large netlist, its primitive input signals $\mathbf{V_{in}}(t)$ and its primitive output signals $\mathbf{V_{out}}(t)$. Application of *node tearing* on the target netlist creates N subcircuits, each one with a number of devices equal to d_i, where $i = 1, 2, ..., N$. *Workload tearing* is the temporal partitioning of the primitive input set based on Equation 1, where **rect** (t) is the rectangular pulse function [16] and T is the time step.

$$\mathbf{V_{in}}(t) = \sum_{j=1}^{M} \mathbf{V_{in}}(t) \ \ \mathbf{rect}\left(t - j\frac{T}{2}\right) \tag{1}$$

Given the time step T, we create M "slices" of the large netlist's primitive inputs (addends of Equation 1). Node and workload tearing create a two dimensional set of intermediate simulations (see Figure 1). Each one is uniquely identified by the pair (i, j) as the simulation of subcircuit i, for the time slice j. Each set member is submitted to a SPICE instance, which runs on a single UE. In order to calculate $\mathbf{V_{out}}(t)$, we require $N \times M$ intermediate SPICE simulations. These simulations require small amounts of memory due to reduced device inventory (d_i) and input signal duration (T). These small memory portions are returned to the multi-processor system after the respective SPICE instance is finished. *This is a major differentiator from SotA and commercial tools, which require strictly increasing amounts of memory.*

A directed, acyclic graph of dependencies between subcircuits indicates the flow of results across intermediate simulations. Signals that are required by an intermediate simulation originate either from primitive inputs (a single time slice) or from the results of a previously executed intermediate simulation. The hypervisor dispatches intermediate simulations and reconciles their outputs based on the interdependency graph. It remains dormant unless an intermediate simulation is completed and another has to initiate. Depending on the platform, the hypervisor initiates SPICE instances by invoking either `fork` or `ssh`. Finally, a data miner creates $\mathbf{V_{out}}(t)$ by combining its time slices in an overlap-save fashion.

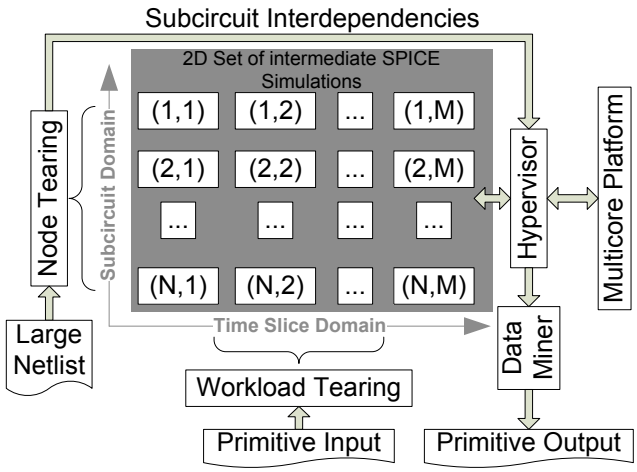


Fig. 1: The proposed hypervisor draws from a set of intermediate SPICE simulations to create the primitive outputs.

B. Illustrative Example

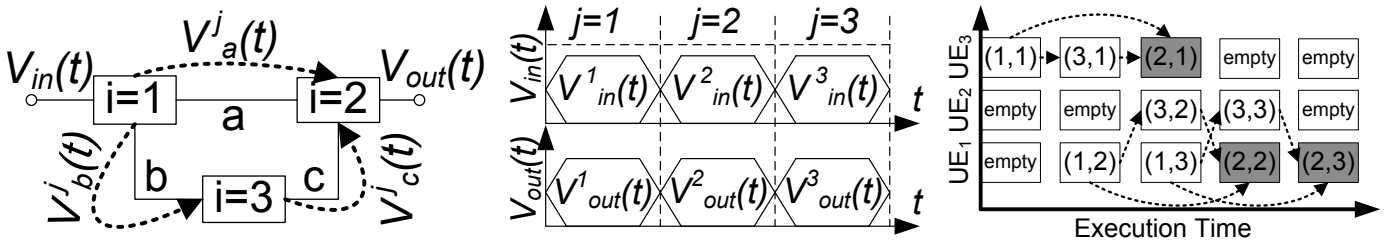
Assume the netlist of Figure 2a. It can be partitioned into three subcircuits, each one being uniquely identified by its i value (dashed arrows show signal dependencies). The netlist needs to be simulated over the extended workload $V_{in}(t)$. With workload tearing, we partition the input vector into three time slices (see Figure 2b). In this simulation, we seek to calculate the primitive output $V_{out}(t)$, which will be assembled per time slice as well. For each time slice, we notate $V_{in}^{j}(t)$ and $V_{out}^{j}(t)$, where $j = 1, 2, 3$. In our example, the voltage signals of each node a, b and c will also be created per time slice, hence we can use the notations $V_a^j(t)$, $V_b^j(t)$ and $V_c^j(t)$.

The set of independent simulations includes the (i, j) pairs, namely a simulation of the i subcircuit over time slice j. Given three available UEs, the hypervisor completes the entire set, by handling intermediate simulations with resolved dependencies (Figure 2c). At the beginning of the execution only simulation $(1, 1)$ has resolved dependencies and can be executed. The dashed arrows of Figure 2c indicate the flow of results between intermediate simulations. For example, simulation $(2, 1)$ can only initiate when simulations $(1, 1)$ and $(3, 1)$ have forwarded their results. Once simulation $(2, 1)$ has been completed, we get the first time slice of the primitive output, namely $V_{out}^{1}(t)$. Times slices $V_{out}^{2}(t)$ and $V_{out}^{3}(t)$ will come from simulations $(2, 2)$ and $(2, 3)$ respectively (gray-shaded boxes in Figure 2c).

IV. Experimental Verification

A. Tested Platforms, Benchmark Netlist & Workloads

We have tested the proposed framework on three representative platforms (see Table I). The first platform is the Single-Chip Cloud Computer (SCC) experimental processor, which is



(a) Node tearing produces the dependency graph (b) Workload tearing and impact on $V_{out}\,(t)$ (c) Mapping of intermediate simulations to UEs

Fig. 2: Members of the (i, j) set are mapped to UEs as execution time progresses and results are forwarded accordingly.

Platform	UE Information	#UEs	Hypervisor	Invocation
SCC	P54C @533MHz	48	@ MCPC*	`ssh`
∼okeanos	QEMU Virt. CPU Version 1.0 @2.1GHz	4	@ 1 UE	`fork`
Xeon	X3470@2.93GHz	4	@ 1 UE	`fork`

* MCPC: Management Console Personal Computer of the SCC

TABLE I: Details of the inspected multicore platforms

Approach	Max. Speedup	Max. #Devices	Inspected Platforms
[10]*	N/A	6	2-Processor System
[11]	[1] × 5.96	19,995	SGI Challenge (12 CPUs)
[12]	[2] × 2.09	96	Intel 2.66GHz PC
[14]	[3] × 11	27,995**	Xilinx Virtex-6 LX760
[15]	[4] × 3.07	7,682	NVIDIA GeForce
Our's	[5] × 2.35	3,670,016	SCC, Cloud VMs, Xeon

[1] vs. single core execution, [2] vs. Fiduccia-Mattheyses partitioning, [3] vs. SPICE on Intel i7-965, [4] vs. Intel Core 2 Quad Core, [5] vs. HSPICE -mt 4
* Only an example with three inverters is presented, ** Size of netlist matrix.

TABLE II: Comparison of the proposed transient SPICE simulation framework with works from the SotA

a 48-core "concept vehicle" created by Intel Labs as a platform for many-core software research [17]. We have also tested our framework on virtual machines (VMs) of the cloud service ∼okeanos [18]. Finally, an Intel®Xeon® server has been used to complete the set of representative test cases. All experiments have been performed using an open source SPICE version called `ngspice` [19]. As *reference for comparison*, we use the commercial tool HSPICE from Synopsis. We choose an execution with four threads running on a similar 2.33GHz Intel®Xeon® machine (flag `-mt 4`).

The benchmark that we have chosen is an array multiplier composed of parallel multiplier cells ("X" boxes of Figure 3a) [20]. We define as *data block size* the length of the multiplier's operands in bits (both operands are assumed of the same length). We increase the device inventory of the netlist by increasing the data block size, reaching beyond 10^6 devices (see Figure 3b). The duration of all workloads is 800ns and the time step T used for workload tearing is 20ns. The selected benchmark allows easy node tearing into identical subcircuits.

B. Simulation Results

For the various sizes of the multiplier array, we apply the same workload both to our framework and the reference tool. The processing time that is exclusively dedicated to circuit simulation can be seen in Figure 3c). For small device sizes, the commercial baseline outperforms our framework. However, beyond an array data block size of 32, HSPICE is unable to deliver a result due to insufficient system memory. At the same time, our proposed framework keeps producing results, even for a netlist with more than 10^6 MOSFET devices. In Figure 3d we can see the execution time occupied by the hypervisor's execution (dispatching SPICE instances, reconciling results).

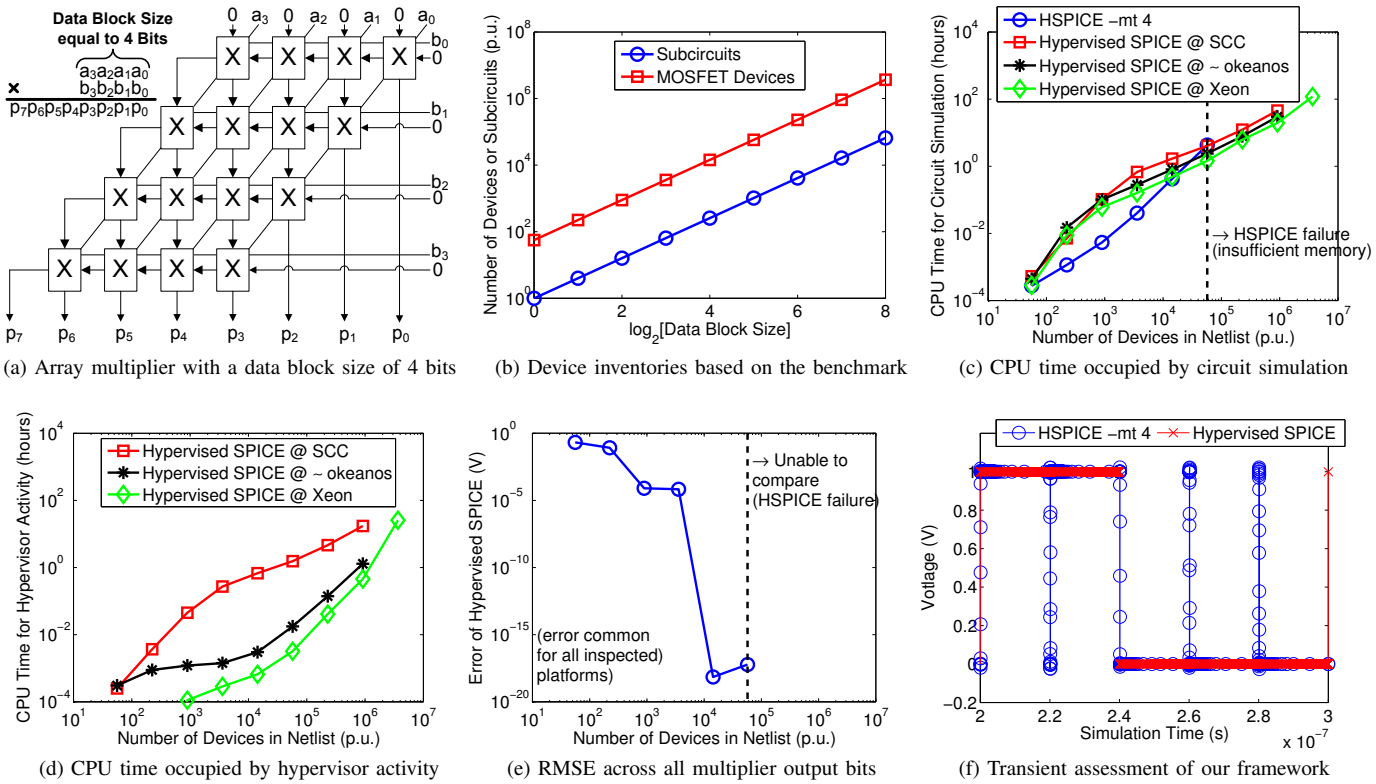
We also assess the accuracy of our approach, by calculating the root mean squared error (RMSE) across all output bits for each multiplier size (see Figure 3e). Errors can be calculated only as long as the reference execution (HSPICE `-mt 4`) does not run out memory (up to a data block size of 32 bits).

In Figure 3f we present a transient output excerpt of our proposed framework and of the reference tool. We can see that the output signals are reproduced very accurately by our simulation framework. Signal transitions and intermediate jitter are the main causes of output error. Voltage droops can be followed accurately by avoiding signal partitioning at time instances where voltage transitions occur. Signal overshoots can be realistically reproduced by appending the appropriate capacitances to the output nodes of each subcircuit that occupies an intermediate simulation.

Having presented experimental results, we can revisit the comparison to the SotA with the three key metrics of Table II. We can safely claim that the device inventories that we address are very much increased in comparison to the SotA, whereas the proposed framework is not limited to specific hardware. Finally, we can identify a $\times 2.35$ speedup in comparison to the reference tool (normalized in cycles), for a data block size of 32 bits (Figure 3c). In general, reduced speedup is to be expected, since our primary goal is the viability of massive, transient SPICE simulations from a memory point of view.

V. Conclusion

The technique proposed in this paper goes further than existing SPICE parallelization approaches by adding a partitioning of the workload on top of existing node tearing. Workload and node partitioning create a set of small SPICE simulations, each one having reduced memory requirements. Being independent, these simulations can be data-partitioned across the UEs of a multi-core platform. With our simulation framework, the demand for main memory is not performed massively and concurrently to the executing platform, as observed in the SotA and even commercial tools. By scaling the initially large SPICE simulation, we avoid hitting the memory constraints of all inspected platforms. We successfully simulate netlists with



(a) Array multiplier with a data block size of 4 bits

(b) Device inventories based on the benchmark

(c) CPU time occupied by circuit simulation

(d) CPU time occupied by hypervisor activity

(e) RMSE across all multiplier output bits

(f) Transient assessment of our framework

Fig. 3: Performance and accuracy evaluation of the hyeprvised SPICE concept proposed in this paper

over 10^6 devices, while the commercial reference fails even in smaller netlists, due to extreme memory requirements. We have integrated our framework in three platforms, covering the range of cloud service infrastructure, regular multi-core systems and advanced chips with an increased number of integrated processors. Thus, we substantiate the reusability of our approach across computing systems, which are usually found in academia or industry.

REFERENCES

[1] F. N. Najm, *Circuit Simulation*, 1st ed. Hoboken, New Jersey, US: Wiley-IEEE Press, 2010.

[2] L. W. Nagel and D. Pederson, "Spice (simulation program with integrated circuit emphasis)," EECS Department, University of California, Berkeley, Tech. Rep. UCB/ERL M382, Apr 1973.

[3] M. Toledano-Luque, et al., "Response of a single trap to ac negative bias temperature stress," in *Reliability Physics Symposium (IRPS), 2011 IEEE International*, april 2011, pp. 4A.2.1 –4A.2.8.

[4] D. Rodopoulos, et al., "Time and workload dependent device variability in circuit simulations," in *IC Design Technology (ICICDT), 2011 IEEE International Conference on*, may 2011, pp. 1 –4.

[5] S. Fuller and L. Millett, "Computing performance: Game over or next level?" *Computer*, vol. 44, no. 1, pp. 31 –38, jan. 2011.

[6] L. W. Nagel, "Spice2: A computer program to simulate semiconductor circuits," Ph.D. dissertation, EECS Department, University of California, Berkeley, 1975.

[7] A. Sangiovanni-Vincentelli, et al., "An efficient heuristic cluster algorithm for tearing large-scale networks," *Circuits and Systems, IEEE Transactions on*, vol. 24, no. 12, pp. 709 – 717, dec 1977.

[8] F. Wu, "Solution of large-scale networks by tearing," *IEEE Transactions on Circuits and Systems*, vol. 23, no. 12, pp. 706 – 713, dec 1976.

[9] E. Lelarasmee, "The waveform relaxation method for time domain analysis of large scale integrated circuits: Theory and applications," Ph.D. dissertation, EECS, University of California, Berkeley, 1982.

[10] F. Wei et al., "Transmission line inspires a new distributed algorithm to solve the nonlinear dynamical system of physical circuit," in *ICCIT 2010*, 30 2010-dec. 2 2010, pp. 816 –821.

[11] N. Frohlich, V. Glockel, and J. Fleischmann, "A new partitioning method for parallel simulation of vlsi circuits on transistor level," in *Design, Automation and Test in Europe Conference and Exhibition 2000. Proceedings*, 2000, pp. 679 –684.

[12] X. Zhou, Y. Wang, and H. Yang, "Dccb and scc based fast circuit partition algorithm for parallel spice simulation," in *ASIC, 2009. ASICON '09. IEEE 8th International Conference on*, oct. 2009, pp. 1247 –1250.

[13] M. Rewieński, "A perspective on fast-spice simulation technology," in *Simulation and Verification of Electronic and Biological Systems*, P. Li, L. M. Silveira, and P. Feldmann, Eds. Springer, 2011, pp. 23–42.

[14] N. Kapre, "Spice2 – a spatial parallel architecture for accelerating the spice circuit simulator," Ph.D. dissertation, California Institute of Technology – Pasadena, California, 2010.

[15] K. Gulati, et al., "Fast circuit simulation on graphics processing units," in *Proceedings of ASP-DAC 2009*, Piscataway, NJ, USA: IEEE Press, 2009, pp. 403–408.

[16] A. Kumar, *Signals and Systems*, 2nd ed. PHI Learning Pvt. Ltd., 2012.

[17] J. Howard, et al., "A 48-core ia-32 processor in 45 nm cmos using on-die message-passing and dvfs for performance and power scaling," *IEEE JSSC*, vol. 46, no. 1, pp. 173 –183, jan. 2011.

[18] V. Koukis, "~okeanos: Delivering iaas to the greek academic and research community," Aug. 2012, VHPC 2012, Rhodes, Greece.

[19] P. Nenzi and H. Vogt, "Spice (simulation program with integrated circuit emphasis)," Tech. Rep. UCB/ERL M382, July 2012.

[20] N. Weste and K. Eshraghian, *Principles of CMOS VLSI design: a systems perspective*, ser. VLSI systems series. Addison-Wesley, 1985.