ADAMIN: Automated, Accurate Macromodelling of Digital Aggressors for Power and Ground Supply Noise Prediction

Zhe Wang[1], *Rajeev Murgai*[2] *and Jaijeet Roychowdhury*[1]

[1]Dept. of ECE, Univ. of Minnesota, Minneapolis, MN 55455

[2]Fujitsu Laboratories of America, Inc., Sunnyvale, CA

Preprint: To appear in IEEE Trans. CAD

Abstract

Estimating interference from large digital blocks and its effect on on-chip power distribution networks is extremely important in deep sub-micron digital and mixed-signal IC design, especially for SoCs. In this paper, we present *automated extraction techniques* that can be used to generate families of small, time-varying macromodels of digital cell libraries from SPICE-level descriptions. Our Automated Digital Aggressor Macromodelling for Interference Noise (**ADAMIN**) approach is based on importing and adapting the Time-Varying Padé (TVP) method, for linear time-varying (LTV) model reduction, from the mixed-signal macromodelling domain. Our approach features naturally higher accuracy than previous ones, and in addition, offers the user a tradeoff between accuracy and macromodel complexity. Extracted macromodels capture a variety of noise interference mechanisms, including IR and $L\frac{dI}{dt}$ drops for power rails. Using **ADAMIN** as a core, it is expected that library characterization methodologies will evolve to include extracted, accurate-by-construction interference noise macromodels for digital cell blocks. Experimental results indicate speedups of several orders of magnitude over full SPICE-level circuits, with prediction accuracies considerably superior to those from commonly-used current-source-based aggressor models.

I. Introduction

In digital and mixed-signal circuit design today, *interference noise* has become of serious concern. This phenomenon is caused by current transients due to the synchronized switching of thousands or millions of digital gates. As illustrated in Figure 1, which depicts a simplified SoC, these injected currents and associated voltage fluctuations travel from digital *aggressors* through power/ground lines, as well as through the chip's substrate, to *victim* circuits on the chip. Victims can include digital circuitry, as well as sensitive analog/mixed-signal circuitry, on the chip.

It is widely expected that the interference noise problem will become of even greater concern in the future. With device technology advances, more and more transistors are packed on single chips, resulting in increased switching

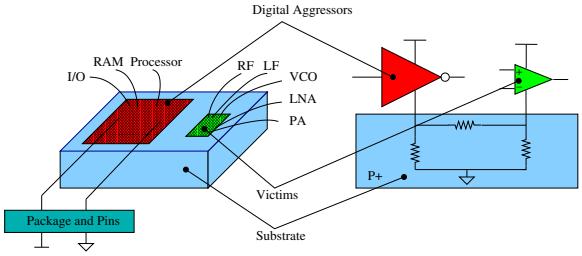

Fig. 1. A simplified illustration of system-on-a-chip design. Large blocks of switching digital circuits ("aggressors") interfere with circuit operation through two main mechanisms. The currents drawn by the aggressors cause voltage fluctuations on power and ground lines because of parasitic resistances and inductances. These voltage fluctuations can cause delays and faulty operation in "victim" digital and unprotected analog circuits. In addition, currents injected into the semiconductor substrate can travel long distances and affect sensitive analog circuitry on the same chip.

currents being drawn into the same physical area. Faster switching leads to increasing inductive noise in power supply grids, while falling supply voltages also lead to reduced noise margins.

To assess and circumvent such problems prior to fabrication, reliable analysis of interference noise needs to be conducted during the design and verification process. There are two major components to this task: extracting appropriately accurate models of power grid and substrate parasitic networks, and estimating the currents injected by digital aggressors blocks into these parasitic networks. As illustrated in Figure 2, it is only by simulating the interactions of the aggressor-injected currents on the parasitic networks that reliable estimates of interference noise can be obtained. Note that the currents injected by the digital aggressors are dependent on the parasitic networks

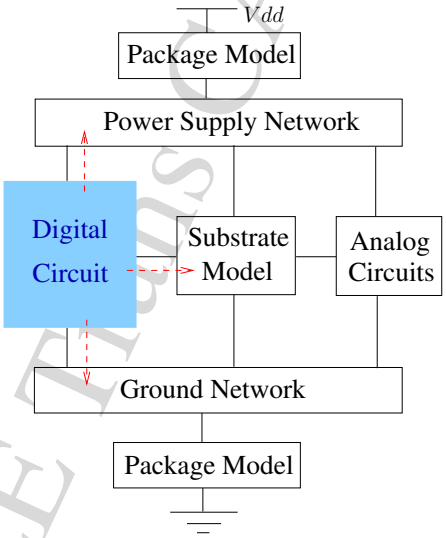

Fig. 2. The switching activities of digital circuits continuously introduce noise (shown by dashed lines) to the power supply network, ground network and substrate. Through these connections, noise impacts the performance of other circuits on the same chip and even the aggressor circuits themselves.

they are connected to, since the interference they cause (*e.g.*, noisy supply voltages) feeds back and modifies their own operation.

While there has been considerable interest and activity in the parasitic extraction problem for power grids, there appear to have been few structured attempts to produce smaller, simpler models of large aggressor clusters that reproduce their aggregate interference effects to reasonable accuracy, yet are many orders of magnitude faster than SPICE-level simulation (a brief review of relevant prior approaches is provided in Section II). This is perhaps not surprising, since the nonlinearity, switching characteristics and complex interconnectivity of digital blocks make them considerably less amenable to the *purely linear* techniques that have so far constituted the mainstay of model-reduction methods. As noted further in Section II, existing approaches for creating aggressor macromodels, useful and critically important as they indeed are, nevertheless suffer from limitations. Important aspects of interference macromodel creation are often manual and rely on ad-hoc heuristics, making it difficult, time-consuming and error-prone to generate them for every technology and design style variant. Partly due to this reason, and also because the underlying mathematical abstractions used can fail to capture nonlinear effects adequately, the accuracy of interference noise prediction using existing aggressor macromodels can vary widely. Indeed, to our knowledge, there has been no work to date on *algorithmic* methods that generate, starting from SPICE-level circuit descriptions, a series of simpler digital aggressor models suitable for power supply network evaluation.

In this paper, we propose and demonstrate a novel approach towards creating sufficiently accurate, yet fast-to-evaluate, interference-noise macromodels of digital aggressors. The key concept of our approach, dubbed **ADAMIN** (Automated Digital Aggressor Macromodelling for Interference Noise), is to use *linear time-varying* (LTV) abstractions to capture important aspects of digital switching nonlinearity, and to generate these small LTV macromodels non-manually, *i.e.*, via appropriate *algorithms*, starting from SPICE-level descriptions of digital cells. LTV abstractions are especially appropriate for modelling digital aggressors, as they capture the impact of switching activity well. Indeed, as pointed out in Section II, **ADAMIN**-generated interference macromodels may be considered to subsume, as special cases, the current-source and ideal-switch-based interference macromodels in common use today. Furthermore, LTV abstractions are amenable to automated model-reduction techniques that provide mathematical metrics of fidelity, such as matching moments of time-varying transfer functions. To this end, **ADAMIN** adapts ideas from the area of nonlinear RF/mixed-signal macromodelling, particularly the Time-Varying Padé (TVP) method of [1], suitably modified and applied to digital aggressors.

The fundamental concept behind **ADAMIN** is to view each digital aggressor block as a box with *small-signal interference "inputs" and "outputs"* connecting it to the supply grid and other external parasitic networks relevant to interference propagation. Large-signal connections, such as logic inputs and outputs, clocks, *etc.*, which determine switching behaviour, will not usually be considered to be amongst these small-signal inputs and outputs, unless interference on those lines is of interest. The critical impact of such large-signal switching is captured by the *time*

variation that they induce in the linear time-varying approximation used by **ADAMIN**. Given a specific time-sequence of logic inputs to the aggressor block, **ADAMIN** generates a small LTV macromodel that approximates the time-varying transfer function between the small-signal inputs and outputs. As such, the macromodel generated is specific to the particular sequence of logic inputs, or *test vector*, chosen.

A key problem in interference estimation is the selection of appropriate test vectors that best represent average or worst-case interference during typical operation.**ADAMIN** does not address this problem directly; instead, the automated macromodel generation capability provided by **ADAMIN** complements test vector selection techniques, since **ADAMIN** can generate accurate interference macromodels for any given test vector. Selection of good representative test vectors is a non-trivial problem, complicated not only by the combinatorial explosion of input possibilities in large digital circuits, but also by the complex technology and circuit-structure dependent analog interactions that come together to determine interference noise. Although **ADAMIN** is orthogonal to the test-vector selection process at the logic propagation level, it provides an excellent means to capture and quickly estimate technology and circuit-dependent aspects of interference noise. Therefore, combined tightly with statistical or worst-case test-vector identification approaches at the logic and digital timing level, we expect that **ADAMIN** will prove itself to computationally useful in helping to filter out and "synthesize", in a hierarchical manner, promising test vectors for worst-case interference in a given extracted design.

Although not central to the present work, we also anticipate that further development of **ADAMIN** to generate macromodels that are *parametrized* with respect to test vector timing, together with integration with static timing analysis and other statistical/worst-case interference vector identification techniques, will enable effective hierarchical library methodologies to address the totality of the interference noise problem. In its present state, it is expected that **ADAMIN** will already be of significant practical use in industrial practice, serving to generate much more accurate drop-in replacements for the simpler models currently used. Note that heuristics for worst-case test excitations are typically already present in such methodologies; the same vectors can be used for generating **ADAMIN** macromodels that are much more accurate.

ADAMIN features several attractive properties. Being based on algorithms and an underlying mathematical formulation (*i.e.*, LTV approximations of the full system) appropriate for the interference problem, **ADAMIN** enables bottom-up extraction at a level of accuracy and automation significantly superior to prior approaches. Macromodel generation via **ADAMIN** does not depend on a-priori knowledge of surrounding supply networks. **ADAMIN** automatically incorporates second order and post-extraction effects in the aggressors, especially important for accuracy in interference noise prediction. Though used in this work for supply noise prediction, simple extensions to **ADAMIN** can produce a *single* interference macromodel capturing all interference mechanisms, including substrate coupling, if the mechanisms are modelled in the full circuit. As such, **ADAMIN** constitutes a sustainable core technique applicable to new devices and technologies of the future. Finally, **ADAMIN** generates macromodels

incrementally in size and accuracy; this makes it possible to utilize size-vs-accuracy tradeoffs with the macromodels it generates.

In this paper, we apply **ADAMIN** to estimating simultaneous switching noise (SSN) and IR drops in power supply networks. Our results confirm speedups of several orders of magnitude even for relatively small blocks, delivering accuracies well under 20%, and typically around 5%. With such characteristics, **ADAMIN** demonstrates promise as a basic core for enabling an accurate, fast, general, sustainable and easily adaptable methodology for interference noise prediction, requiring relatively small changes to cell characterization methodologies, but enabling generation of bottom-up accurate-by-construction models for power supply and substrate noise analysis.

Though our experiments in this paper use SPICE-level transient simulation to evaluate speedups from ADAMIN-generated macromodels, it is likely that similar speedups will result in other simulators (*e.g.*, recent "fast" simulators such as HSIM, NanoSim, or switch-level simulators such as ATTSIM, PowerMill, etc). This is due to the fact that a key aspect of digital system structure, *i.e.*, the large-scale repetition of blocks, remains substantially similar when full blocks are replaced by their ADAMIN interference macromodels. Since each ADAMIN macromodel simulates much faster than the original SPICE-level circuit, speedup and accuracy benefits stemming from ADAMIN-generated macromodels may be expected across a variety of simulation engines.

The remainder of the paper is organized as follows. We first provide a brief review of previous work in Section II. In Section III, we describe **ADAMIN**. Application examples and simulation results are presented in Section IV.

II. Review of Previous Approaches towards Digital Aggressor Macromodelling

In principle, full SPICE-level circuit representations of digital aggressors could be used to capture nonlinear switching and interference behavior in detail, but this is impractical because the numbers of transistors in modern digital circuits can easily reach tens or even hundreds of millions. Therefore, smaller and faster-to-evaluate macromodels for estimating the interference of digital blocks have been sought. By replacing SPICE-level representations of digital blocks with corresponding macromodels, and simulating them together with power grid and substrate parasitics, useful estimates of interference can be obtained in practice.

As indicated earlier in Section I, two major interference transmission mechanisms are transmission through the power supply grid (IR drops and $L\frac{dI}{dt}$ noise) and transmission through the substrate (capacitive coupling from aggressors to the substrate, followed by distributed propagation through a resistive/capacitive substrate). Most previous approaches to developing aggressor macromodels have tended to specialize for one mechanism or the other. In this paper, we focus on work relevant to the supply grid interference problem.

Dharchoudhury et al [2] proposed an interference macromodel for power/ground networks consisting of independent, ideal current sources for each cell. The current source waveform is obtained by simulating the SPICE-level circuit description with a perfect power supply voltage and storing the current value at every time point in a

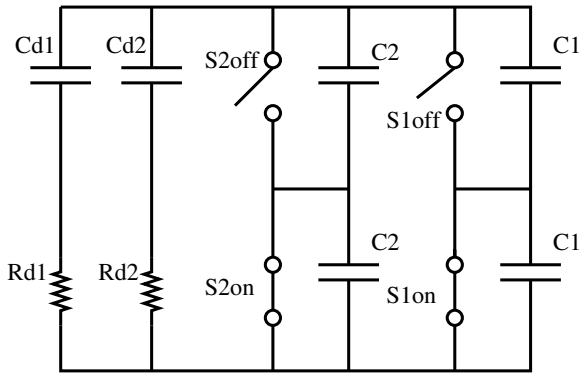

Fig. 3. Equivalent switching circuit proposed by Chen and Ling in [4].

vector. This approach approximates the nonlinear problem by a simple linear one and makes possible the analysis of large blocks of logic, unaffordable using full SPICE-level circuits. Ideal current source macromodels are, however, a very simple approximation that is unable to capture loading effects of the surrounding networks. Not capturing the impact of power supply voltage variations on the macromodel's current can introduce significant errors; for example, power supply voltage swings of 10% can result in transient current swing changes of 30%. The ease with which generation of such models can be automated is a great attraction of this approach, which has enjoyed widespread adoption within the industry. A similar approach was proposed for capturing substrate noise injection by van Heijningen et al [3].

A different approach towards interference macromodelling was proposed by Chen and Ling [4]; as depicted in Figure 3, their macromodel for digital blocks utilizes capacitors controlled by ideal switches (with on/off resistances) to imitate the switching behavior of digital cells. Aspects of this macromodel constitute an improvement over current source macromodels, in that the current supplied by the macromodel can change depending on the loading of the power/ground network. However, this macromodel can also be less accurate than the current-source macromodel; for example, when a perfect supply network is connected to it, the waveform predicted by the current-source macromodel is more accurate, having been derived from full SPICE-level simulation of the block. Chen and Ling's macromodel, which is based on templates of capacitors and switches, can be considered to be essentially a manual macromodelling approach aided by simulations and heuristics, relying as it does on human understanding of the circuit block and its switching characteristics.

It is instructive to note how the **ADAMIN**-generated macromodels of this paper subsume these prior approaches for aggressor macromodelling, building on their positive features while circumventing their disadvantages. To imitate the switching behavior of digital cells, the approach exemplified by [4] uses capacitors switched by ideal switches. Switched capacitors (and resistors) constitute a very simple *linear time-varying* model; **ADAMIN** can therefore be considered to be an algorithmic extension of this approach, which not only produces more refined time-varying models, but also overcomes the other disadvantages of [4], notably the large-network problem for big blocks.

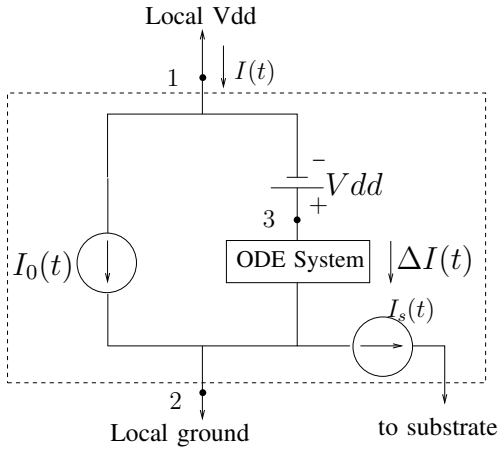

Fig. 4. **ADAMIN** macromodel of a digital cell.

The simpler but more automated and more popular approach of [2], which employs an ideal current source to represent each block, also constitutes a (trivial) simplification of an **ADAMIN**-generated macromodel: referring to Figure 4, the current source of [2] is simply the **ADAMIN**-macromodel with the small-signal LTV network omitted.

III. **ADAMIN**: AUTOMATED DIGITAL AGGRESSOR MACROMODELLING FOR INTERFERENCE NOISE

The objective of **ADAMIN** is to "extract" a small macromodel, useful for predicting supply interference, from a detailed description of the digital aggressor block. Since interference is a highly analog phenomenon, often depending on details such as the transient loading, drive and charge storage characteristics of the digital block, a natural starting point for **ADAMIN** is the full SPICE-level circuit description of the digital block. Such descriptions are typically heavily used during cell library design and characterization, hence are easily available. The main utility of **ADAMIN** lies in its ability to automatically abstract good interference models from these descriptions, a task that is very challenging to perform manually.

SPICE-level circuit descriptions are, in essence, systems of *nonlinear differential-algebraic equations* (DAEs). A general system of DAEs, suitable for circuits, that represents the digital aggressor block (*e.g.*, see Figure 5) under consideration is

$$\begin{aligned} \frac{d}{dt}\, q(x(t)) + f(x) + b(t) + Bu(t) = 0, \\ z(t) = c^T x(t). \end{aligned} \tag{1}$$

In (1), $x(t)$ represents a size-n vector of unknowns; $q(\cdot)$ and $f(\cdot)$ are vector nonlinear functions representing charge/flux and current terms, respectively; $b(t)$ is a vector of external input or forcing signals (such as logic and clock inputs for digital blocks). $u(t)$ and $z(t)$ are small-signal interference noise inputs and outputs to the system, respectively; for the purposes of **ADAMIN**, they represent variables describing the block's connections to the (external) supply grid. The size n of the vectors in (1) is expected to be large, since (1) represents a large

block consisting of many digital gates. The utility of the form of (1) is that it encompasses any kind of circuit or system, subsuming all technologies, circuit topologies and design styles of interest.

In order to capture switching nonlinearities with linear time-varying abstractions, **ADAMIN** adapts ideas from the area of nonlinear RF/mixed-signal macromodelling. In particular, the Time-Varying Padé (TVP) method [1], suitably modified for the digital aggressor context, is applied. Appendix A contains a detailed description of the TVP algorithm; in this section, we describe its use and utility.

The first step in **ADAMIN** is to simulate the full circuit description (*i.e.*, (1)) at the SPICE level, using transient analysis (*e.g.*, [6], [7]) or periodic steady-state analysis (*e.g.*, [8]–[12]) for one or a few clock cycles. This simulation is performed for a given vector of logic inputs, using nominal or ideal values of the power supply connected to the block, as captured in $b(t)$; information about the surrounding supply network is not necessary for this simulation and subsequent macromodel extraction. Once the simulation has been carried out, detailed waveforms for all nodes and branch currents are available. It is also necessary to store and access a number of simulator-internal matrices (see (2) below); for this reason, access to the internals of a well-structured circuit simulator is required for **ADAMIN** to be implemented.

Once the transient solution is available, (1) is linearized [1] in a time-varying fashion around this solution[1] to obtain

$$\begin{aligned} \frac{d}{dt}\left[C(t)y(t)\right] + G(t)y(t) + Bu(t) = 0, \\ z(t) = c^T y(t). \end{aligned} \tag{2}$$

(2) represents the LTV approximation of the full nonlinear system (1), with $y(t)$ representing linearized deviations from the full nonlinear solution. The sizes of the matrices and vectors in (1) and (2), are identical, corresponding to the size of the full aggressor block.

TVP [1] essentially converts (2) into a system of similar form but of much smaller size, which nevertheless replicates the linear time-varying transfer function of (2) to within a desired accuracy. It achieves this by applying efficient Krylov-subspace-based numerical techniques (*e.g.*, [13]–[15]) to reduce (2) to the form

$$\begin{aligned} -T_q \frac{d}{dt} w(t) + w(t) = R_q u(t), \\ z(t) = l_q^T(t)\, w(t). \end{aligned} \tag{3}$$

In (3), $w(t)$ represents the *reduced* state vector of the macromodel; it is of size $q \ll n$. T_q is a *small* square matrix of size q that captures the system's dynamics, while R_q and l_q are block-vectors (*i.e.*, rectangular matrices with as many size-q columns as there are inputs and outputs in $u(t)$ and $z(t)$, respectively) that capture the connections of the inputs and outputs, respectively, to the system. This macromodel, depicted in Figure 4, is used as a replacement for the original for interference noise prediction purposes in **ADAMIN**.

[1]For simplicity of exposition, we illustrate the time-varying *linear* case here; weakly nonlinear extensions (*e.g.*, [1]) can also be applied for greater accuracy.

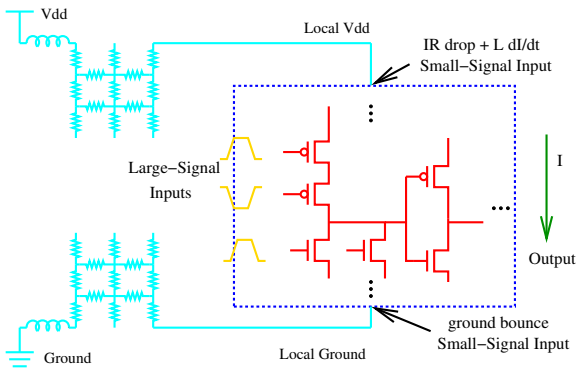

Fig. 5. In the LTV digital aggressor macromodelling process, logic inputs of digital cell are defined as large-signal inputs and the noise in the power and ground line are defined as small-signal inputs.

A key step in creating useful digital aggressor noise macromodels via **ADAMIN** is to designate the "inputs" $u(t)$ and the "outputs" $z(t)$ of (1), (2) and (3) appropriately. It is the *deviations* from nominal behaviour of the power, ground and substrate interfaces that constitute the inputs and outputs of interest. For example, as depicted in Figure 5, the input may be taken to be the voltage noise at the block's local VDD and ground nodes, with the output the resulting changes in current drawn; or vice-versa. The main utility of the macromodel is to approximate the transfer function between inputs and outputs well. With the use of multiple input/output ports (*i.e.*, $u(t)$ and $z(t)$ are small vectors), a single accurate macromodel can be used to capture power grid interference, substrate coupling and other interactions. Because the simple aggressor models can be embedded with power grid and substrate parasitics and the entire network simulated as a whole, the methodology captures feedback and bidirectional interactions between the aggressors and the supply network.

Once the reduced system (3), *i.e.*, the ADAMIN-generated interference macromodel, is available, it is used to replace the original circuit (1). The external supply network's equations are added to augment (3), creating a single interacting system that can be simulated speedily to capture interference interactions between the aggressor and the supply network. The importance of the "input" and "output" variables $u(t)$ and $z(t)$ lies in that they constitute the interface between the aggressor macromodel and the supply network. For example, with (3) representing the boxed digital aggressor in Figure 6, the modified nodal equations of the reduced system (including the simple supply network) become

$$\begin{cases} \frac{V_1(t)-V_{dd}}{R} + I_0(t) + I_d(t) = 0 \\ u(t) = V_1(t) - V_{dd} \\ -\boldsymbol{T}_q \frac{d\boldsymbol{x}}{dt} + \boldsymbol{x}(t) = \boldsymbol{R}_q u(t) \\ I_d(t) = \boldsymbol{l}_q(t)\boldsymbol{x}(t) \end{cases} . \tag{4}$$

Interference/SSN noise at any point in the circuit can then be obtained by simulating this small ODE system. Any simulation technique, whether time or frequency domain, that is applicable to differential equation systems

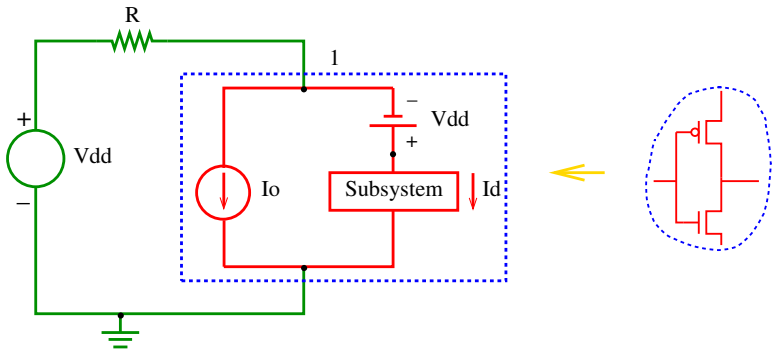

Fig. 6. **ADAMIN**-reduced circuit of a digital aggressor block connected to Vdd through a resistor.

may be used; in this work, we use SPICE-like transient simulation, but the reduced system may equally well be simulated in fast simulation engines such as timing-level simulators, HSIM, NanoSim, *etc.*, for additional speed gains.

A number of features of the above procedure make **ADAMIN** particularly suitable as a core component of interference prediction methodologies:

- **Automated bottom-up generation:** In contrast to today's ad-hoc manual approaches for aggressor noise macromodel generation, **ADAMIN** is based on **algorithms** that, starting from SPICE-level descriptions of gates and digital blocks, extract small time-varying macromodels. This makes it possible to employ it as the core of a highly automated, convenient-to-apply interference prediction methodology.
- **Correct underlying mathematical representations: ADAMIN** generates general linear time-varying aggressor noise macromodels. These abstractions are appropriate for accurate representation of nonlinear switching systems that feature moderate swings at some inputs and outputs, such as power, ground, and substrate contact nodes. When **ADAMIN**-generated models are embedded in parasitic networks, they capture the bi-directional impact of noise feedback on the injected currents correctly. The time-variation can be parametrized so that shifts in clock inputs to a block can be incorporated easily without re-generation of the macromodel. Further, macromodels generated are in the form of a linear or weakly nonlinear time invariant block followed by memoryless time variation, making incorporation in existing tools and methodologies simple.
- **Modular extraction independent of surrounding environment:** Unlike some prior approaches that rely on simulating aggressor blocks within the parasitic environment that they inject noise into, **ADAMIN** is modular, requiring no a-priori knowledge of the surroundings the aggressor will be embedded in. As a result, aggressor macromodels can be extracted off-line at the same time as cell library characterization is performed, and stored as integral parts of the cell library. The macromodels can then be used as drop-in replacements for the original blocks for the purpose of interference noise prediction. As high-performance design typically starts from SPICE-level analysis of cells during library characterization, this approach fits well into cell characterization

methodologies and design processes.

- **Single unified aggressor model for all interference mechanisms:** Also in contrast to prior approaches, macromodels generated by **ADAMIN** can capture all interference mechanisms of interest. These include, but are not limited to, $L\frac{dI}{dt}$ *and* IR *drops* in power supply networks, the major noise injection mechanisms for today's technologies. **ADAMIN** is easily extensible to other interference mechanisms such as substrate noise.
- **Smooth accuracy vs evaluation speed tradeoff: ADAMIN** incrementally generates macromodels of increasing size and accuracy, making it possible to obtain a family of models featuring a smooth accuracy/speed tradeoff. In addition, the resolution of time variation can also be changed. It should be noted that one feature of the interference noise problem is the relatively relaxed levels of prediction accuracy needed - accuracies within 20% are considered good - compared to macromodels for, *e.g.*, mixed-signal system-level simulation. While **ADAMIN** is capable of producing very accurate macromodels if needed, this additional flexibility can be exploited for generating particularly fast-to-evaluate ones tailored to interference noise prediction.
- **Second-order effects; applicability to new devices, technologies and device styles: ADAMIN** automatically takes into account second-order effects, directly from SPICE-level MOSFET models such as BSIM3/4. Being based on algorithms operating on general systems of differential-algebraic equations, **ADAMIN** applies unchanged to any kind of new technology or design style. In particular, it is expected that **ADAMIN** will be valuable for predicting interference in nanotechnological systems of the future.

IV. Experimental Results

In this section, we first apply **ADAMIN** to a 2000-inverter digital block with an imperfect power supply network. We apply a known voltage noise to the local Vdd and compare current waveforms in order to verify the correctness of our model. Next, we assess **ADAMIN** on a 4-stage carry chain adder block, comprising gates with multiple logic inputs. We then apply **ADAMIN** to analyse interactions between the digital aggressor and the supply network, by embedding the 2000-inverter digital block within a simple power grid model. Finally, we simulate four 2000-inverter blocks together with a power supply network.

As mentioned earlier, **ADAMIN** requires access to simulator internals. For our experiments, we have developed and used a prototype circuit simulator, written primarily in MATLAB and with modules in C/C++, for implementing **ADAMIN** as well as for simulation of the full circuits and their extracted macromodels. This simulator includes commonly-used analysis capabilities familiar from SPICE [6], [7], including DC and transient analyses; it also includes a number of RF analyses, such as steady-state and envelope simulation methods. At this point, the simulator lacks support for BSIM MOS models, hence all simulations in this paper were carried out using a smoothed version of SPICE's Level 1 MOS model, *i.e.*, based on a smoothed, continuous version of the Schichman-Hodges equations. Note that **ADAMIN** automatically applies to any kind of model supported by the simulator in which it is implemented.

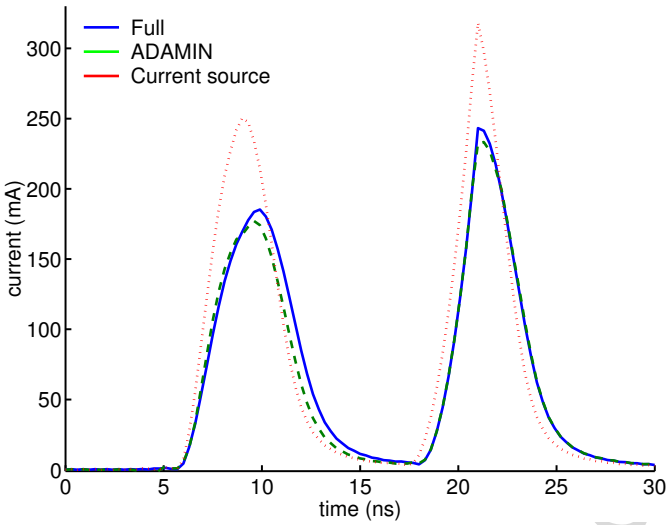

Fig. 7. Comparison of currents drawn by a full 2000-inverter block (Level 1 MOSFET model), ADAMIN-generated macromodel and a current-source macromodel [2].

A. *2000 Inverter Digital Block: basic validation of current perturbations*

For basic validation and to illustrate features of ADAMIN, we first apply a DC voltage perturbation of 0.2V to the power supply line (which has a nominal Vdd of 2V, chosen as an example) via a perturbing voltage source[2]. The order of the reduced macromodel is chosen to be $q = 2$; this value of q brings the macromodel's computational complexity to about the same level as that in [2]. The accuracy of the two, however, is markedly different, as shown in Fig 7. When the power supply voltage drops by 10% from its ideal value, the peak current drops more than 30%. Notice that the current computed using **ADAMIN**'s macromodel closely matches the full circuit's both in peak value and slope. The former is important in for IR drops while the latter is critical for $L\frac{dI}{dt}$ effects. The **ADAMIN** macromodel speeds up the simulation by about 60, while the error of the estimated current is only 3.84% compared to the SPICE model.

B. *Carry-chain Adder Block: basic validation*

Figure 9 depicts a 4-stage carry-chain adder block that adds two 4-bit numbers; the internal structure of each block, consisting of 12 MOS devices, is also shown. Once again, an explicit power supply perturbation (from a nominal VDD of 1.8V in this case) of 0.2V is provided for the simulation, and the current drawn by the block is considered. Each carry-chain block was macromodelled separately by ADAMIN (using size $q = 3$), using the logic drive specific to the block. The macromodels were then assembled to represent the 4-block circuit. Figure 9 depicts the time-varying currents predicted by macromodel-based simulation, vs those by the full circuit. It can

[2]To capture time variation, we choose uniform step sizes of 300ps ($\frac{1}{100}$ of the clock period) for simplicity; in general, the time step will be determined dynamically during the model-generation process by the transient simulation process.

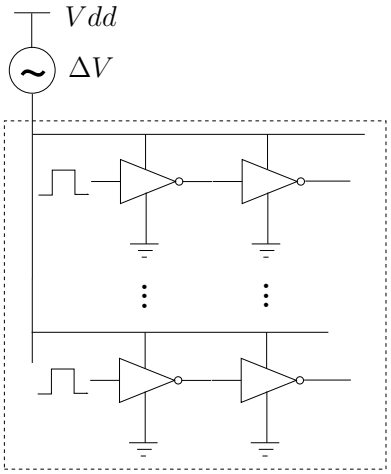

Fig. 8. A block consists of 2000 inverters connected to a noisy power supply.

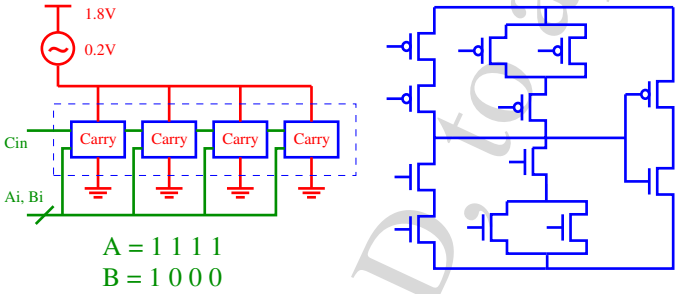

Fig. 9. Carry chain adder block.

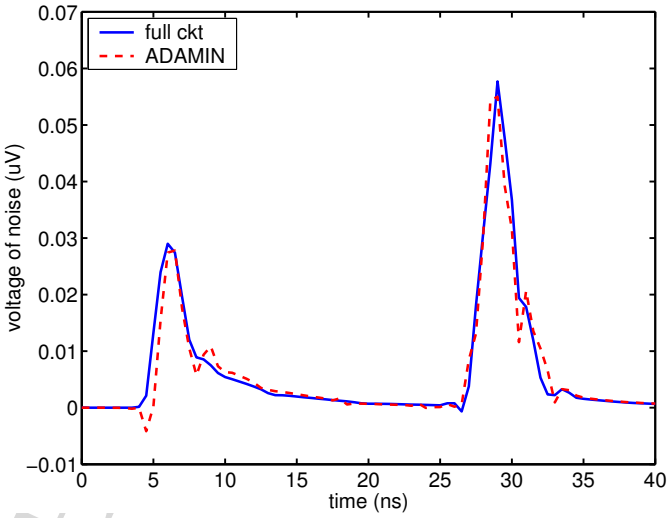

Fig. 10. Simulation of full circuit vs ADAMIN-generated macromodel for carry-chain adder circuit.

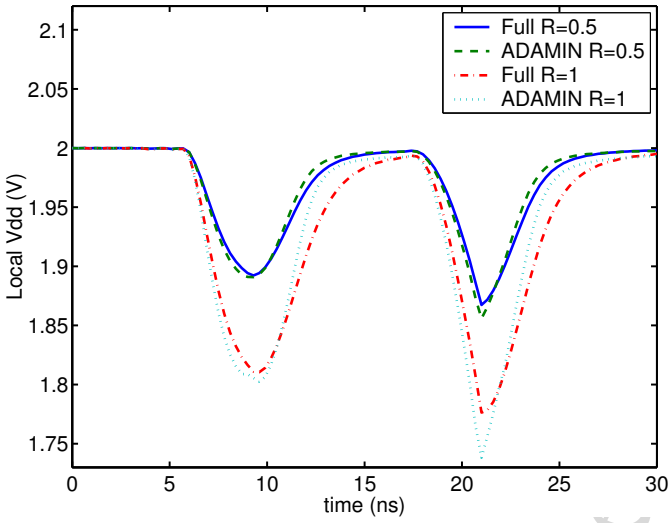

Fig. 11. Comparison of local Vdds using **ADAMIN** -generated macromodel and full circuit; resistive power grid.

be seen that despite the small size of the macromodel, there is reasonable agreement between the two waveforms. Even for this small circuit, the speedup afforded by the macromodel is well over two orders of magnitude.

C. 2000-Inverter with Resistive/Inductive Power Grid

Next, the 2000-inverter digital block is simulated together with a simple power supply network model consisting of a single resistor. The ADAMIN-generated macromodel obtained from the first example was used for the digital block. With this setup, the main advantage of ADAMIN-generated macromodels, *i.e.*, their interaction with the supply grid network, can be assessed. Vdd noise generated for two values of the supply resistor are shown in Figure 11, with the ADAMIN-generated macromodel compared against the full circuit. The interference noise prediction error due to the macromodel spans the range 8.63% through 16.99%; such accuracies are typically considered entirely adequate in interference noise estimation.

Next, the impact of *inductive* supply grid effects was simulated using ADAMIN-generated macromodels of two sizes ($q = 2$ and $q = 20$), the full circuit, and a current-source macromodel. As can be seen, the $q = 2$ macromodel is already considerably superior to the current-source in terms of accuracy; for $q = 20$ (still representing a size reduction of about a 100 over the original circuit), the macromodel's prediction is almost identical to that of the full circuit, correctly capturing both voltage drops and overshoots due to the inductive component.

A speedup of about $5600\times$ was obtained by using the **ADAMIN**-generated macromodels.

D. Multiple Blocks with Staged Delays and Power Grid

Finally, as shown in Figure 13, an 8000-inverter digital circuit is replaced with four 2000-inverter macromodels as extracted for the first example. The supply network's resistive parasitics are also shown in Figure 13. The

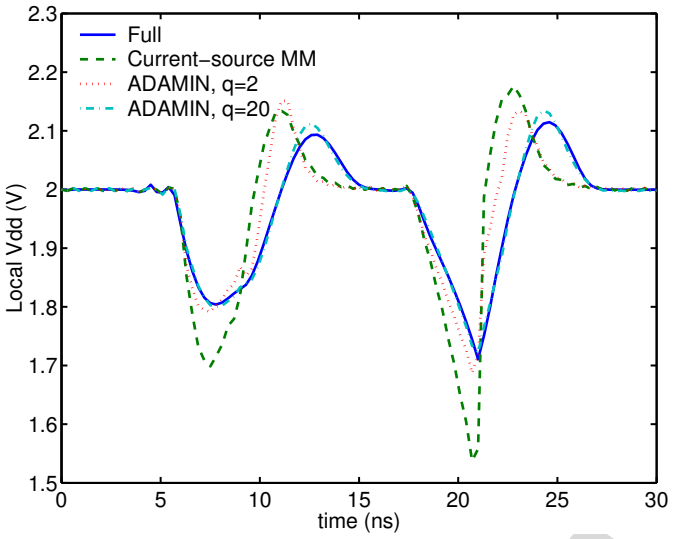

Fig. 12. Comparison of local Vdds using **ADAMIN** -generated macromodel and full circuit; resistive/inductive power grid.

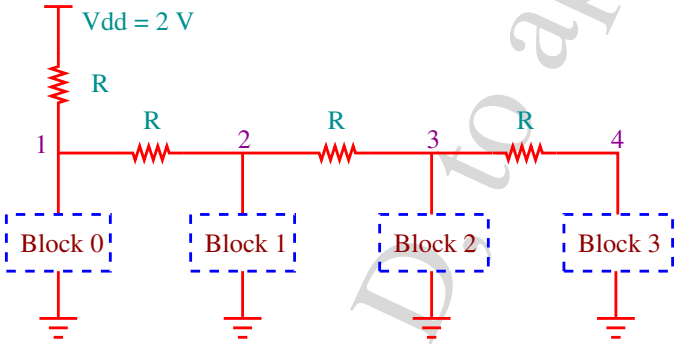

Fig. 13. Replace the digital blocks of a 8000-inverter circuit with four macromodels of 2000-inverter-block with different delays.

logic inputs to the four blocks are staggered in time with delays representing each block's rise and fall times. The currents drawn by each block are shown in Figure 14, while the combined effect of the currents on the supply network's voltage, as delivered to the blocks, is plotted for each block in Figure 15. It was not practical to simulate the full circuit in our MATLAB-based prototyping framework; however, extrapolating speedups obtained from the 2000 inverter block, we estimate a speedup of about 17000x for the $q = 20$ macromodel.

V. Conclusions

In this paper, we have presented **ADAMIN**, an automated digital aggressor macromodelling technique based on linear time-varying representations and Krylov-subspace-based model-order reduction. Using the TVP automated macromodelling algorithm as its core, **ADAMIN** captures switching variations in large blocks with small, algorithmically-generated macromodels. As a result, **ADAMIN** macromodels are well suited for assessing the reliability of power distribution networks. Experimental results indicate speedups of over 4 orders of magnitude with accuracies well within 20%.

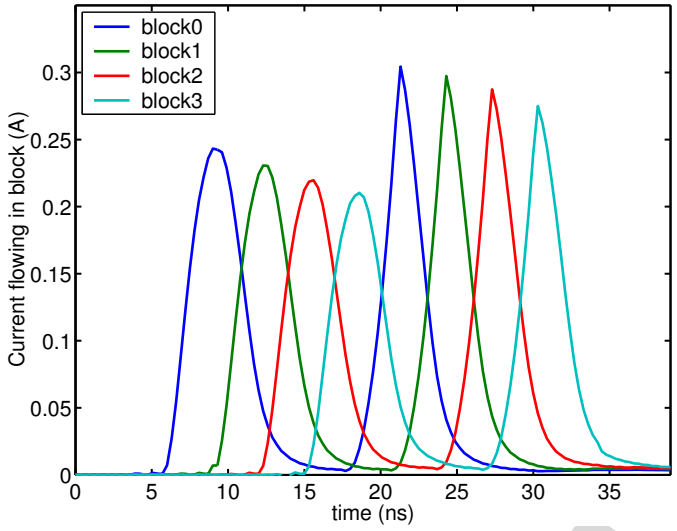

Fig. 14. Current drawn by each macromodel.

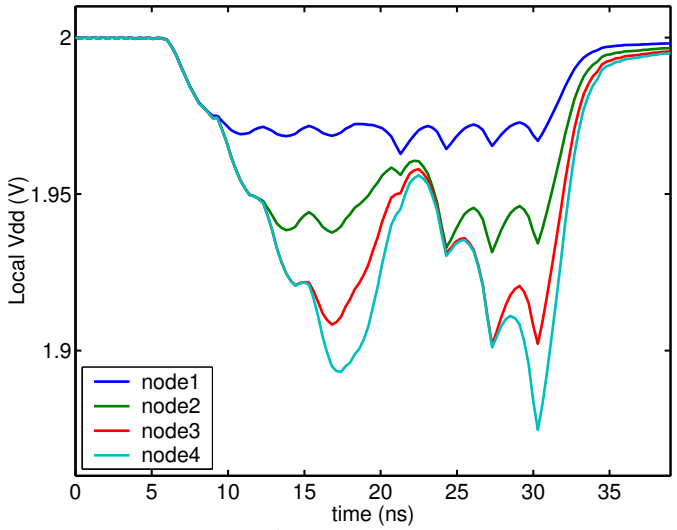

Fig. 15. Local Vdd at each macromodel.

We are currently extending **ADAMIN** by developing methods for hierarchical compaction of LTV macromodels, together with techniques for parameterizing the timing of test vectors applied to the aggressors. We expect these additions to tie together the extraction of interference estimation macromodels at every level, and to enable deriving whole-chip interference noise analysis in a automated, bottom-up manner with easily-controlled accuracy/speed tradeoffs.

Acknowledgments

This work was supported by NSF grants CCR-0204278 and CCR-0312079, DARPA grant number SA0302103, and by the SRC CADT program, using resources and computational facilities from the Minnesota Supercomputing

Institute and the Digital Technology Center of the University of Minnesota.

REFERENCES

[1] J. Roychowdhury, "Reduced-order modeling of time-varying systems," IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing, vol. 46, no. 10, pp. 1273–1288, October 1999.

[2] A. Dharchowdhury, R. Panda, D. Blaauw, R. Vaidyanathan, B. Tutuianu, and D. Bearden, "Design and analysis of power distribution networks in PowerPC^{TM} microprocessors," in Proc. of IEEE DAC, Anaheim, CA, June 15-19, 1998, pp. 738–743.

[3] M. v. Heijningen, M. Badaroglu, S. Donnay, M. Engels, and I. Bolsens, "High-level simulation of substrate noise generation including power supply noise coupling," in Proc. of IEEE DAC, Los Angeles, CA, June 5-9, 2000, pp. 738–743.

[4] H. H. Chen and D. D. Ling, "Power supply noise analysis methodology for deep-submicron VLSI design," in Proc. of IEEE DAC, Anaheim, CA, June, 1997, pp. 638–643.

[5] K. L. Shepard and D. J. Kim, "Static noise analysis for digital integrated circuits in partially-depleted silicon-on-insulator technology," in Proc. of IEEE DAC, Los Angeles, CA, June 5-9, 2000, pp. 239–242.

[6] L. Nagel, "SPICE2: a computer program to simulate semiconductor circuits," Ph.D. dissertation, EECS Dept., Univ. Calif. Berkeley, Elec. Res. Lab., 1975, memorandum no. ERL-M520.

[7] T. Quarles, "Analysis of Performance and Convergence Issues for Circuit Simulation," Ph.D. dissertation, EECS Dept., Univ. Calif. Berkeley, Elec. Res. Lab., Apr. 1989, memorandum no. UCB/ERL M89/42.

[8] T. Aprille and T. Trick, "Steady-state analysis of nonlinear circuits with periodic inputs," Proc. IEEE, vol. 60, no. 1, pp. 108–114, Jan. 1972.

[9] K. S. Kundert and A. Sangiovanni-Vincentelli, "Simulation of nonlinear circuits in the frequency domain," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. CAD-5, no. 4, pp. 521–535, Oct. 1986.

[10] R. Gilmore and M. Steer, "Nonlinear circuit analysis using the method of harmonic balance – a review of the art. Part I. Introductory concepts." Int. J. on Microwave and Millimeter Wave CAE, vol. 1, no. 1, 1991.

[11] R. Melville, P. Feldmann, and J. Roychowdhury, "Efficient multi-tone distortion analysis of analog integrated circuits," in Proc. IEEE CICC, May 1995, pp. 241–244.

[12] J. Roychowdhury, "Analysing circuits with widely-separated time scales using numerical PDE methods," IEEE Trans. Ckts. Syst. – I: Fund. Th. Appl., May 2001.

[13] Y. Saad, Iterative methods for sparse linear systems. Boston: PWS, 1996.

[14] P. Feldmann and R. Freund, "Reduced-order modeling of large linear subcircuits via a block Lanczos algorithm," in Proc. IEEE DAC, 1995, pp. 474–479.

[15] A. Odabasioglu, M. Celik, and L. Pileggi, "PRIMA: passive reduced-order interconnect macromodelling algorithm," IEEE Trans. CAD, pp. 645–654, Aug. 1998.

[16] ——, "Prima: passive reduced-order interconnect macromodeling algorithm," in IEEE/ACM International Conference on Computer-Aided Design, San Jose, CA, Nov 9-13, 1997, pp. 257–260.

[17] Y. Saad, Iterative Methods for Sparse Linear Systems. PWS-Kent, Boston, 1996.

[18] D. L. Boley, "Krylov space methods on state-space control models," Circuits, Systems and Signal Processing, vol. 13, pp. 733–758, 1994.

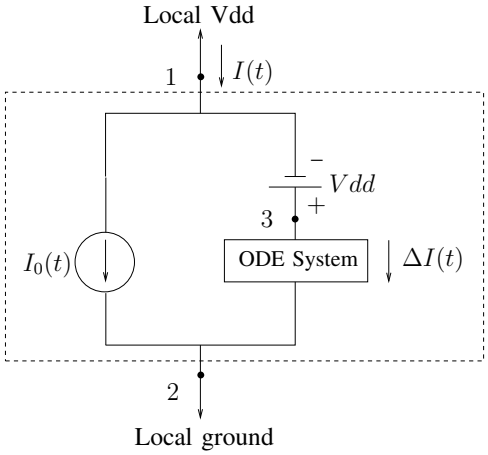

Fig. 16. **ADAMIN** macromodel of a digital cell for current estimation.

APPENDIX

A. TVP - Time-Varying Padé

TVP was originally developed for mixed-signal/RF/analog circuit macromodelling applications, such as mixers and switched-capacitor filters. However, as adapted in **ADAMIN**, TVP is applicable for the analysis digital interference problems, *i.e.*, reduction of large digital logic blocks for SSN and IR drop prediction purposes.

By separating the time scales of the small input $u(t)$ and the logic input $\boldsymbol{b}_l(t)$, (1) can be re-expressed in MPDE [12] form as:

$$\begin{aligned} \frac{\partial \boldsymbol{q}(\hat{\boldsymbol{y}})}{\partial t_1} + \frac{\partial \boldsymbol{q}(\hat{\boldsymbol{y}})}{\partial t_2} + \boldsymbol{f}(\hat{\boldsymbol{y}}(t_1,t_2) &= \boldsymbol{b}_l(t_1) + \boldsymbol{b}u(t_2) \\ \hat{z}_t(t_1,t_2) = \boldsymbol{d}^T\hat{\boldsymbol{y}}(t_1,t_2), \quad & z_t(t) = \hat{z}_t(t,t), \end{aligned} \tag{5}$$

where the hatted variables are bivariate (i.e., two-time scales) forms of the corresponding variables in (1). It can be shown [12] that any solution of (1) generates a solution of (5).

Solving (5) with $u(t_2) = 0$ and denoting the solution by $\hat{\boldsymbol{y}}^*(t_1)$, the output is given by $z_t(t) = \hat{z}_t(t,t) = \boldsymbol{d}^T\hat{\boldsymbol{y}}^*(t)$. Note that $z_t(t)$ is $I_0(t)$ in Figure 16. By linearizing (5) around $\hat{\boldsymbol{y}}^*(t_1)$, a linear, time-varying MPDE form can be obtained:

$$\begin{aligned} \frac{\partial(\boldsymbol{C}(t_1)\hat{\boldsymbol{x}})}{\partial t_1} + \frac{\partial(\boldsymbol{C}(t_1)\hat{\boldsymbol{x}})}{\partial t_2} + \boldsymbol{G}(t_1)\hat{\boldsymbol{x}} &= \boldsymbol{b}u(t_2), \\ \hat{z}(t_1,t_2) = \boldsymbol{d}^T\hat{\boldsymbol{x}}(t_1,t_2), \quad & z(t) = \hat{z}(t,t). \end{aligned} \tag{6}$$

Vectors $\hat{\boldsymbol{x}}$, $\hat{z}$, and z are the small-signal versions of $\hat{\boldsymbol{y}}$, $\hat{z}_t$ and z_t, respectively; $\boldsymbol{C}(t_1) = (\partial \boldsymbol{q}(\hat{\boldsymbol{y}})/\partial \hat{\boldsymbol{y}})|_{\hat{\boldsymbol{y}}^*(t_1)}$ and $\boldsymbol{G}(t_1) = (\partial \boldsymbol{f}(\hat{\boldsymbol{y}})/\partial \hat{\boldsymbol{y}})|_{\hat{\boldsymbol{y}}^*(t_1)}$ are time-varying matrices. Eq. (6) implies that the bivariate output $\hat{z}(t_1,t_2)$ is linear in the small input signal $u(t_2)$, but this linear relationship changes depending on the value of the system time t_1. To obtain the time-varying transfer function from $u(t_2)$ to $\hat{z}(t_1,t_2)$, we apply the Laplace transform to (6) with

respect to t_2. Since $\boldsymbol{C}(t_1)$ is independent of t_2, we have $\partial \boldsymbol{C}(t_1)/\partial t_2 = \boldsymbol{0}$. It then follows that

$$\frac{\partial(\boldsymbol{C}(t_1)\hat{\boldsymbol{X}}(t_1,s))}{\partial t_1} + s\boldsymbol{C}(t_1)\hat{\boldsymbol{X}}(t_1,s) + \boldsymbol{G}(t_1)\hat{\boldsymbol{X}}(t_1,s) = \boldsymbol{b}U(s) \qquad \hat{Z}(t_1,s) = \boldsymbol{d}^T\hat{\boldsymbol{X}}(t_1,s), \tag{7}$$

where s denotes the Laplace variable along the t_2 time axis, and the capital symbols denote transformed variables.

We now collect samples of $\boldsymbol{C}(t_1)$, $\boldsymbol{G}(t_1)$, $\hat{\boldsymbol{X}}(t_1,s)$, and $\hat{Z}(t_1,s)$ over $t_1 \in [0,T_1]$, at a total of $N+1$ instances $\{t_{1,n}\}_{n=0}^{N}$ with $t_{1,0}=0$, and $t_{1,N}=T_1$. In the following, we consider the case where the system is periodic in t_1, and take T_1 to be one period of the system.[3] Eq. (7) can then be re-expressed in a differential form as

$$\begin{aligned} \frac{\boldsymbol{C}_{(t_{1,n})}\hat{\boldsymbol{X}}_{(t_{1,n},s)} - \boldsymbol{C}_{(t_{1,n-1})}\hat{\boldsymbol{X}}_{(t_{1,n-1},s)}}{\delta_n} + [s\boldsymbol{C}(t_{1,n}) + \boldsymbol{G}(t_{1,n})]\hat{\boldsymbol{X}}(t_{1,n},s) = \boldsymbol{b}U(s) \\ \hat{Z}(t_{1,n},s) = \boldsymbol{d}^T\hat{\boldsymbol{X}}(t_{1,n},s), \quad \forall n \in [1,N], \end{aligned} \tag{8}$$

where $\delta_n = t_{1,n} - t_{1,n-1}$. Notice that at each snapshot $t_{1,n}$, $\hat{\boldsymbol{X}}(t_{1,n},s)$ still consists of m unknowns (i.e., the number of node voltages and branch currents of the system), while $\boldsymbol{C}(t_{1,n})$ and $\boldsymbol{G}(t_{1,n})$ are matrices of corresponding dimension. Stacking such vectors, we construct super vectors: $\bar{\boldsymbol{X}}(s) = [\hat{\boldsymbol{X}}^T(t_{1,1},s),\ldots,\hat{\boldsymbol{X}}^T(t_{1,N},s)]^T$, $\bar{\boldsymbol{B}} = \boldsymbol{1}_{N,1} \otimes \boldsymbol{b}$, where $\mathbf{1_{N,1}}$ is a N by 1 all-one vector, and $\otimes$ denotes Kronecker product. Correspondingly, we also construct block matrices: $\boldsymbol{\mathcal{G}} = \mathrm{diag}\{\boldsymbol{G}(t_{1,1}),\ldots,\boldsymbol{G}(t_{1,N})\}$, $\boldsymbol{\mathcal{C}} = \mathrm{diag}\{\boldsymbol{C}(t_{1,1}),\ldots,\boldsymbol{C}(t_{1,N})\}$, $\boldsymbol{\mathcal{D}} = \boldsymbol{I}_N \otimes \boldsymbol{d}$, and $\boldsymbol{\Delta} = (\mathrm{dial}\{1/\delta_1,\ldots,1/\delta_N\}(\boldsymbol{I}_N - \boldsymbol{J}_N)) \otimes \boldsymbol{I}_m$, where $\boldsymbol{I}_N$ stands for a N by N identity matrix, and $\boldsymbol{J}_N$ a N by N circulant matrix with first column $[0,1,0,\ldots,0]^T$, and first row $[0,\ldots,0,1]$. It can be readily verified that the time-varying transfer function is given by:

$$\boldsymbol{H}(s) = [H(t_{1,1},s),\ldots,H(t_{1,N},s)]^T = \boldsymbol{\mathcal{D}}^T[s\boldsymbol{\mathcal{C}} + \boldsymbol{\mathcal{G}} + \boldsymbol{\Delta}\boldsymbol{\mathcal{C}}]^{-1}\bar{\boldsymbol{B}}, \tag{9}$$

such that $\boldsymbol{H}(s)U(s) = \hat{\boldsymbol{Z}}(s)$ with definition $\hat{\boldsymbol{Z}}(s) = [\hat{Z}^T(t_{1,1},s),\ldots,\hat{Z}^T(t_{1,N},s)]^T$. Notice that the dimension of $\boldsymbol{\mathcal{C}}$ (and also $\boldsymbol{\mathcal{G}}$, $\boldsymbol{\Delta}$) is $mN \times mN$. Once Eq. (9) is obtained, model order reduction techniques can be applied directly. Along the lines of [1], a model of reduced order $q < mN$ can be obtained by casting (9) into the standard form $\boldsymbol{H}(s) = \boldsymbol{\mathcal{D}}^T[\boldsymbol{I}_{mN} - s\boldsymbol{\mathcal{A}}]^{-1}\boldsymbol{\mathcal{R}}$ with definitions $\boldsymbol{\mathcal{A}} = -[\boldsymbol{\mathcal{G}} + \boldsymbol{\Delta}\boldsymbol{\mathcal{C}}]^{-1}\boldsymbol{\mathcal{C}}$ and $\boldsymbol{\mathcal{R}} = [\boldsymbol{\mathcal{G}} + \boldsymbol{\Delta}\boldsymbol{\mathcal{C}}]^{-1}\bar{\boldsymbol{B}}$, and applying Krylov subspace methods [16], [17]. Applying the block Arnoldi algorithm [18], the resultant qth order transfer function that approximates $\boldsymbol{H}(s)$ in (9) is given by

$$\boldsymbol{H}_q(s) = \boldsymbol{L}_q^T[\boldsymbol{I}_q - s\boldsymbol{T}_q]^{-1}\boldsymbol{R}_q, \tag{10}$$

where $\boldsymbol{L}_q = \boldsymbol{V}_q^T\boldsymbol{\mathcal{D}}$ is a $q \times N$ matrix, $\boldsymbol{T}_q$ is a $q \times q$ block-Hessenberg matrix, $\boldsymbol{R}_q = \boldsymbol{V}_q^T\boldsymbol{\mathcal{R}}$ is a $q \times 1$ vector, and $\boldsymbol{V}_q$ is the $mN \times q$ matrix consisting of the q orthogonal bases generated by applying block Arnoldi to $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{R}}$.

Notice that the dimension of the matrix for model order reduction is $mN \times mN$. To make our methodology applicable to large elemental blocks (hundreds to thousands of nodes) with several tens of sampling time points, we choose Krylov-subspace techniques because of their $\boldsymbol{O}(N)$ complexity. Alternative SVD-based techniques can

[3] For more general cases, and frequency domain treatments, the reader is referred to [1].

be proven to be more optimal, but their $\boldsymbol{O}(N^3)$ complexity makes them less attractive for the interference noise problem.

From (6), the transfer function (10) of order q corresponds to the ODE system in Fig. 16, which translates the noise in power supply grids to its corresponding current change. (10) is easily transformed into the time domain as

$$-\boldsymbol{T}_q \frac{d\boldsymbol{x}}{dt} + \boldsymbol{x} = \boldsymbol{R}_q u(t), \quad z(t) = \boldsymbol{l}_q(t)\boldsymbol{x}(t), \tag{11}$$

where $\boldsymbol{x}$ is a vector of size q, $z(t)$ is the output (that is, $\Delta I(t)$ in our macromodel in Fig. 16), and $\boldsymbol{l}_q(t)$ is the $q \times 1$ time-varying vector that relates the system (states) to the output. To link $\boldsymbol{l}_q(t)$ with the $q \times N$ matrix $\boldsymbol{L}_q$ in (10), we note that the nth column of $\boldsymbol{L}_q$ is identical to $\boldsymbol{l}_q(t_{1,n})$, $\forall n \in [1, N]$.

Summarizing, the macromodel corresponding to any specific digital cell can be uniquely represented by $I_0(t)$ and the ODE system (11) that generates $\Delta I(t)$ given the input voltage noise $u(t)$. Being independent of the power supply variation, $I_0(t)$ can be computed off-line. The other current $\Delta I(t)$, however, relies on the power supply noise, and has to be computed by taking the overall system as a whole. But notice that the ODE system parameters captured in $\boldsymbol{T}_q$, $\boldsymbol{R}_q$, and $\boldsymbol{L}_q$ do not depend on the voltage variation, and can thus be computed off-line, and stored together with $I_0(t)$ in a cell library. The key feature of the LTV macromodel (11) is that it consists of a small number of nodes, as opposed to the original (1). Thus, (11) serves as a low-complexity approximation that can replace (1) in a complete power supply distribution analysis, including package models and on-chip supply networks.