Deep Learning for Spatial Supply Noise Estimation in a Processor Chip

Rusong Weng, Yaguang Li, Wei Gao, Leilei Wang, Xufeng Kou and Pingqiang Zhou
School of Information Science and Technology
ShanghaiTech University, Shanghai, P. R. China
E-mail: {wengrs, liyg, gaowei, wangll, kouxf, zhoupq}@shanghaitech.edu.cn.

***Abstract*—To detect voltage emergency caused by supply voltage fluctuation, people have proposed linear models to estimate the supply noise at a hotspot based on the readings of limited on-chip noise sensors. But such linear model only works well in a power delivery network with linear elements. In this work, we propose to use deep learning to build an accurate noise estimation model for each hotspot in a real chip with nonlinear circuit loads. In our experiments we test the performance of our deep learning based approach on a set of benchmarks with different degree of non-linearity, and we also compare its performance with prior work.**

I. Introduction

As shown in Fig. 1, the power is delivered to each transistor/gate in a chip through a power delivery network (PDN), which can be modeled as an electrical network consisting of resistors (R), capacitors (C) and inductors (L) [1]–[4]. Supply noise, defined as the difference between the real supply

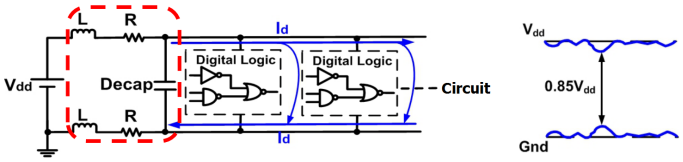

Fig. 1: The source of supply noise in a processor chip.

voltage and the ideal supply V_{dd}, includes the resistance-induced IR drop and the temporal supply variations caused by the interactions among the inductors, capacitors and the time-varying circuit loads I_d. If the noise-caused supply voltage drops below a certain threshold, typically 15% of the V_{dd}, a voltage emergency event happens, which may cause timing errors in the logic circuits, and can potentially result in functional failures.

In recent years, the industry and academia have designed both direct and indirect ways to monitor the supply noise and thus detect the voltage emergency [5]–[11]. For example, Intel has developed both analog [7] and digital [5], [6] sensors to *directly* detect the supply noise at a certain number of critical spots across the chip. Although these sensors can monitor the supply noise around them at the time granularity of clock cycle, they can only achieve partial coverage of the chip due to the very limited number of available sensors. To address this limitation, Liu et al. [8], [9] propose a full-chip noise estimation framework which can calculate the supply noise at any hotspot across the chip, by modeling it as a *linear* function of the sensor readings. To develop such estimation model, they have adopted both statistical and machine learning methods. The main problem with [8], [9] is that the supply noises at different locations in a given chip are not linearly correlated, given the fact that the loads of the chip, such as the transistors, typically have non-linear I-V curves. In addition to directly monitoring the supply noise, Reddi et al. [10] have proposed voltage signature-based method to predict voltage emergency *indirectly* at the architecture level. In their method, they first record all the voltage signatures (that is, micro-architecture events or instructions flows) that have resulted in voltage emergency in the history, then at runtime they can predict the potential voltage emergency based on the recorded voltage signatures.

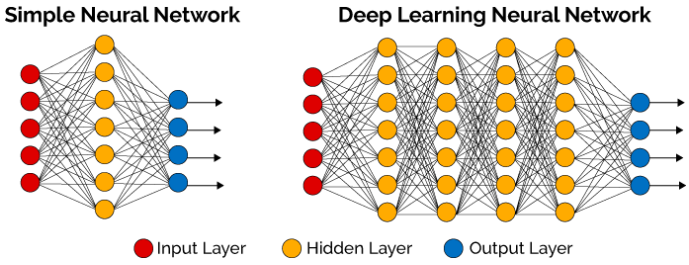

Fig. 2: Illustration of a deep learning network [12].

Deep learning, as one particular kind of machine learning, has already been proved useful in many fields, such as computer vision, machine translation, speech recognition, natural language processing, drug design and board game programs [13]–[15]. A typical deep learning model (as shown in Fig. 2) includes an input layer, one output layer and a proper number of inner hidden layers, the inherent structure that makes deep learning powerful in expressing/generating the implicit correlation in a given data set [13]. In this work, we explore such advantage of deep learning and use it to develop the estimation model of the supply noise for any given hotspot.

II. The PDN with nonlinear circuit loads

As stated in Section I, the PDN can be modeled as a RLC network as shown in Fig. 3(a). The on-chip transistors/gates draw power from the PDN through the access nodes. Excluding the circuit loads, the PDN itself only contains linear RLC components and voltage sources. Traditionally,

978-1-5386-4441-6/18/$31.00 ©2018 IEEE

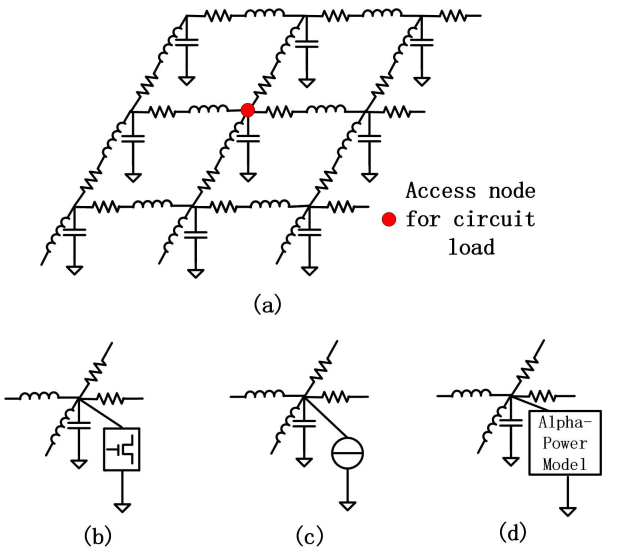

Fig. 3: A simple PDN (a) and its load models: (b) - transistor, (c) - current source model, and (d) - Alpha-power law model.

to simply the analysis of the PDN, the circuit loads (see Fig. 3(b)) are modeled as current sources with piece-wise-linear temporal current (see Fig. 3(c)). However, we know that the I-V characteristic of the circuit loads such as transistors are typically nonlinear, and a simplified linear model may bring big error to the PDN analysis.

In our work, we use the well-known Alpha-power model [16] to describe the I-V characteristic of the transistors (see Fig. 3(d)) as

$$I = \beta \cdot V_{dd}^{\alpha} \tag{1}$$

In this model, $\alpha \in [1, 2]$, and β is a fitting parameter that can be found by SPICE simulations at a given technology node. Eq. (1) clearly shows that the I-V characteristic of the circuit loads is super-linear and in the worst case, can be quadratic.

Given the fact that the loads of the PDN is nonlinear, it is not hard to find that the correlations among the supply voltages at the access nodes, the responses of the PDN system, are nonlinear. If we want to build a noise estimation model by exploring such correlations, we should be able to capture their nonlinear feature in the modeling process.

III. How to build the deep neural network for supply noise estimation?

In our work, we build a deep neural network with one input layer, one output layer, and three hidden layers:

1) For the input layer, its size (number of neurons) equals to the number of noise sensors used in the chip. The inputs to this layer are the readings of the noise sensors.
2) For the first two hidden layers, they have the same number of neurons as the input layer, and each neuron evaluates the following function

$$t = g(W^T A + b) \tag{2}$$

where

- W is the vector of weights (on the layer-to-layer connections in Fig. 2),
- A is the vector of inputs for each layer (in our work, they represent the supply voltage)
- b is the bias vector, and
- g is the activation function. There are several widely used activation functions. In our work, we choose to use the ReLU (rectified linear unit) function given by

$$g(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases} \tag{3}$$

3) The third hidden layer has the same size as the first two hidden layers, the only difference is that its neurons don't implement the activation operations.
4) For the output layer, its size equals to the number of hotspots to be monitored in the chip. The output of this layer is the estimated supply voltage at the hotspots.

When we train the deep neural network, we are solving the optimization problem of

$$\min_{\phi} f(N(\phi, X), Y) \tag{4}$$

while N represents the neural network, ϕ is the set of model parameters (Ws and bs in Eq. (2)), Y is the real supply voltage at the hotspots in the training set and f is the loss function reflecting the difference between the network output and Y.

IV. Experimental Results

A. Experimental setup

In our experiments, we build a test circuit with 9 sensors and 17 hotspots. We then randomly selected 10,000 samples of voltage traces for these sensors/hotspots as the training set and another 114,000 samples as the test set. For the deep neural network, we use the framework Pytorch [17] to train and test the built network. The optimization problem (4) is solved by the Adam (Adaptive Moment Estimation) optimizer [18], a widely-used optimization algorithm for deep learning. We train the deep neural network for multiple times and pick the best one as the final result.

B. The Results

In this section, we compare our model built by deep learning approach with the linear model built by [8] on 19 benchmarks. For fair comparison, we use the same experimental setup (sensor/hotspot set, training/test set, etc.) for these two approaches.

To evaluate the accuracy of the built models, we use the metric of relative error, which is defined as

$$relative\_error = \sqrt{\frac{\sum (predict\_value - real\_value)^2}{\sum real\_value^2}} \tag{5}$$

and we report the average value of relative error for each benchmark in our results.

As stated in Eq. (1), the I-V curve of a circuit load is determined by two key parameters α and β. In a real chip with tens of millions of transistors, the transistors may work in different configurations that result in different values of α and

Benchmark	α									
	1.20		1.43		1.65		1.85		2.00	
	[8]	Ours	[8]	Ours	[8]	Ours	[8]	Ours	[8]	Ours
1	0.125%	0.148%	0.297%	0.257%	0.518%	0.207%	0.728%	0.596%	0.938%	0.225%
2	0.129%	0.177%	0.291%	0.265%	0.469%	0.242%	0.672%	0.588%	0.814%	0.293%
3	0.089%	0.123%	0.173%	0.158%	0.285%	0.214%	0.369%	0.377%	0.481%	0.285%
4	0.148%	0.227%	0.325%	0.376%	0.546%	0.309%	0.765%	0.742%	0.969%	0.492%
5	0.246%	1.553%	0.565%	1.707%	0.986%	1.714%	1.407%	2.381%	1.821%	1.596%
6	0.075%	0.275%	0.129%	0.327%	0.234%	0.383%	0.324%	0.489%	0.439%	0.642%
7	0.080%	0.164%	0.145%	0.238%	0.251%	0.227%	0.315%	0.350%	0.417%	0.368%
8	0.092%	0.173%	0.204%	0.214%	0.320%	0.221%	0.456%	0.425%	0.548%	0.272%
9	0.115%	0.176%	0.245%	0.271%	0.405%	0.239%	0.565%	0.574%	0.715%	0.383%
10	0.087%	0.178%	0.175%	0.266%	0.300%	0.270%	0.402%	0.419%	0.505%	0.383%
11	0.101%	0.174%	0.227%	0.231%	0.361%	0.240%	0.511%	0.431%	0.619%	0.305%
12	0.184%	0.498%	0.411%	0.603%	0.718%	0.588%	1.022%	1.191%	1.292%	0.614%
13	0.079%	0.323%	0.199%	0.358%	0.351%	0.395%	0.564%	0.682%	0.724%	0.692%
14	0.116%	0.248%	0.247%	0.336%	0.438%	0.342%	0.619%	0.713%	0.794%	0.513%
15	0.122%	0.174%	0.267%	0.236%	0.423%	0.211%	0.600%	0.481%	0.710%	0.265%
16	0.133%	0.245%	0.312%	0.355%	0.540%	0.311%	0.777%	0.713%	0.989%	0.364%
17	0.098%	0.157%	0.224%	0.219%	0.350%	0.215%	0.497%	0.429%	0.592%	0.262%
18	0.108%	0.174%	0.246%	0.276%	0.426%	0.236%	0.608%	0.528%	0.761%	0.324%
19	0.102%	0.145%	0.217%	0.232%	0.381%	0.208%	0.529%	0.474%	0.653%	0.309%
Average	0.117%	0.281%	0.258%	0.365%	0.437%	0.356%	0.617%	0.662%	0.778%	0.452%

TABLE I: Relative error with fixed $\beta = 0.6$ and varying α.

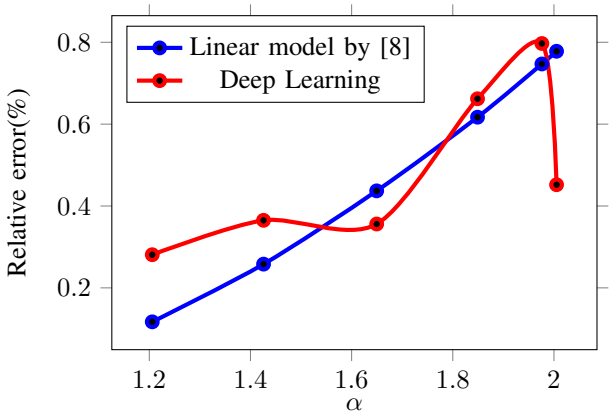

Fig. 4: Average relative error over all benchmarks when $\beta = 0.6$.

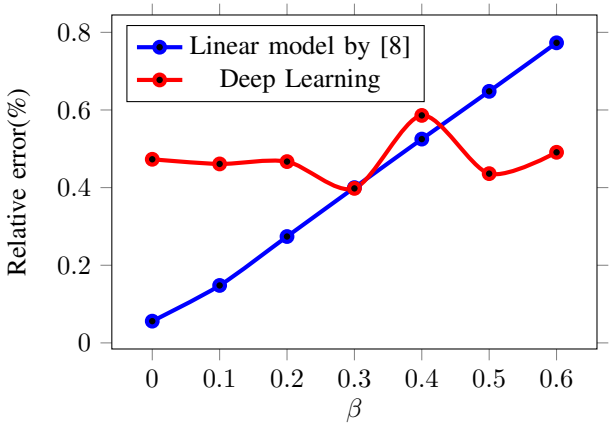

Fig. 5: Average relative error over all benchmarks when $\alpha = 2$.

β. Therefore, in our work, we sweep these two parameters and test the corresponding performance of two estimation models.

Our results are summarized in Table I, Fig. 4, Table II and Fig. 5. The tables list the detailed results for all the 19 benchmarks, while the figures show the aggregated results over all the benchmarks. At first, when we fix β and sweep α from 1.2 to 2, the circuit loads gradually change from close-to-linear elements towards quadratic elements. As shown in Table I and Fig. 4, in this case, the estimation error of the linear model by [8] increases from 0.117% to 0.778%, which is about 6.6X worse. In contrast, our method based on deep learning has the capability of generating nonlinear models to adapt to the changing nonlinear circuit loads, and therefore, achieve a relatively stable estimation accuracy (with only 1.6X worse). Further, when we fix α (=2) and sweep β from 0 to 0.6 to increase the current loads of the PDN, the loads are exerting larger impact on the system and they tends to see larger supply noise. As shown in Table II and Fig. 5, the linear model suffers from an increasing estimation error while our deep learning based method is much more stable. Fig. 6 shows the comparison of two methods at one hotspot when $\alpha = 2$ and $\beta = 0.6$. For this corner case, deep neural network is much better than the linear model.

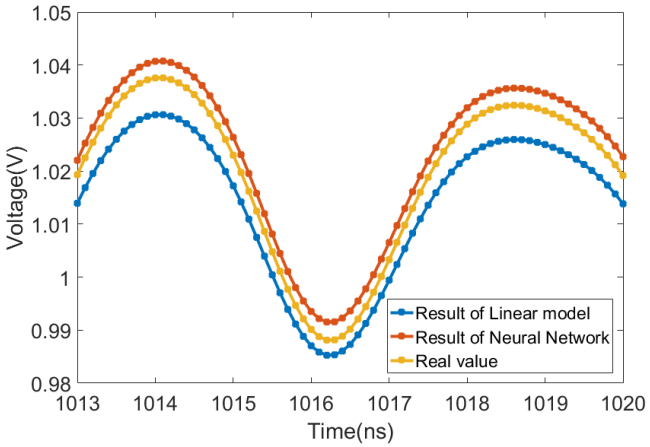

Fig. 6: Predicted supply voltage versus real voltage at one hotspot, with α (=2) and $\beta = 0.6$.

Benchmark	β													
	0		0.1		0.2		0.3		0.4		0.5		0.6	
	[8]	Ours	[8]	Ours	[8]	Ours	[8]	Ours	[8]	Ours	[8]	Ours	[8]	Ours
1	0.066%	0.266%	0.153%	0.270%	0.310%	0.320%	0.459%	0.225%	0.606%	0.505%	0.754%	0.258%	0.923%	0.298%
2	0.047%	0.380%	0.168%	0.369%	0.317%	0.375%	0.458%	0.291%	0.572%	0.522%	0.695%	0.305%	0.812%	0.336%
3	0.038%	0.262%	0.105%	0.246%	0.190%	0.254%	0.263%	0.210%	0.341%	0.379%	0.402%	0.233%	0.463%	0.284%
4	0.067%	0.499%	0.189%	0.486%	0.345%	0.487%	0.507%	0.382%	0.650%	0.638%	0.808%	0.423%	0.969%	0.472%
5	0.090%	1.496%	0.297%	1.568%	0.575%	1.665%	0.862%	1.630%	1.165%	1.974%	1.478%	1.776%	1.807%	1.909%
6	0.086%	0.617%	0.095%	0.576%	0.140%	0.515%	0.199%	0.452%	0.282%	0.499%	0.353%	0.493%	0.443%	0.611%
7	0.044%	0.368%	0.098%	0.345%	0.158%	0.327%	0.219%	0.283%	0.286%	0.343%	0.350%	0.331%	0.421%	0.369%
8	0.042%	0.386%	0.120%	0.356%	0.223%	0.351%	0.319%	0.289%	0.390%	0.422%	0.470%	0.287%	0.546%	0.308%
9	0.048%	0.378%	0.149%	0.357%	0.263%	0.362%	0.378%	0.298%	0.487%	0.494%	0.603%	0.338%	0.714%	0.369%
10	0.057%	0.417%	0.111%	0.391%	0.189%	0.372%	0.271%	0.321%	0.354%	0.404%	0.431%	0.369%	0.510%	0.409%
11	0.048%	0.383%	0.131%	0.356%	0.248%	0.354%	0.356%	0.292%	0.444%	0.427%	0.534%	0.305%	0.614%	0.334%
12	0.046%	0.552%	0.230%	0.576%	0.431%	0.642%	0.634%	0.568%	0.865%	0.922%	1.078%	0.664%	1.289%	0.746%
13	0.064%	0.658%	0.096%	0.621%	0.199%	0.583%	0.307%	0.486%	0.424%	0.642%	0.554%	0.525%	0.700%	0.686%
14	0.080%	0.561%	0.149%	0.535%	0.262%	0.514%	0.388%	0.421%	0.528%	0.640%	0.661%	0.449%	0.803%	0.510%
15	0.035%	0.387%	0.158%	0.365%	0.289%	0.361%	0.417%	0.292%	0.513%	0.438%	0.619%	0.279%	0.709%	0.298%
16	0.076%	0.364%	0.163%	0.366%	0.326%	0.414%	0.484%	0.327%	0.641%	0.610%	0.798%	0.378%	0.975%	0.428%
17	0.046%	0.343%	0.131%	0.326%	0.248%	0.322%	0.355%	0.260%	0.439%	0.408%	0.524%	0.259%	0.590%	0.277%
18	0.042%	0.373%	0.138%	0.359%	0.264%	0.363%	0.384%	0.297%	0.519%	0.457%	0.639%	0.329%	0.751%	0.359%
19	0.043%	0.305%	0.129%	0.290%	0.235%	0.301%	0.335%	0.243%	0.463%	0.411%	0.559%	0.280%	0.647%	0.322%
Average	0.056%	0.473%	0.148%	0.461%	0.274%	0.467%	0.400%	0.398%	0.525%	0.586%	0.648%	0.436%	0.773%	0.491%

TABLE II: Relative error with fixed α (=2) and varying β.

In summary, when the circuit loads are purely or close to linear, or the load current is small, the linear model proposed in [8] performs well in terms of estimation error. However, when the load is gradually switching from linear to quadratic, or the impact of circuit loads on PDN increases, the linear model fails to keep its high estimation accuracy. In contrast, deep learning based method is good at processing the nonlinear components in the systems and it has better adaptivity. Therefore, in a real chip design with many nonlinear elements, deep learning is the preferred method for supply noise estimation.

V. Conclusion

In this paper, we exploit deep learning to build a supply noise estimation model for each hotspot based on noise sensors with analog readings. To reflect the inherent nonlinear feature of the power delivery network, we have adopted the widely used alpha-power law model to characterize the circuit loads. Experimental results on a big set of benchmarks show that, for a circuit with many nonlinear elements, our method has better performance than prior work in terms of estimation accuracy and robustness in presence of load variations.

References

[1] S. S. Sapatnekar and H. Su, "Analysis and optimization of power grids," *IEEE Design Test of Computers*, vol. 20, no. 3, pp. 7–15, May 2003.

[2] M. S. Gupta, J. L. Oatley, R. Joseph, G. Y. Wei, and D. M. Brooks, "Understanding voltage variations in chip multiprocessors using a distributed power-delivery network," in *Conference on Design, Automation and Test in Europe*, 2007, pp. 624–629.

[3] L. Wang and P. Zhou, "Leakage power reduction in multicore chips via online decap modulation," in *China Semiconductor Technology International Conference*, 2016, pp. 1–3.

[4] L. Wang, C. Zhuo, and P. Zhou, "Run-time demand estimation and modulation of on-chip decaps at system level for leakage power reduction in multicore chips," *Integration, the VLSI Journal*, 2018.

[5] C. Lefurgy, A. Drake, M. Floyd, M. Allen-Ware, B. Brock, J. Tierno, J. Carter, and R. Berry Jr, "Active guardband management in power7+ to save energy and maintain reliability," *IEEE Micro*, vol. 33, no. 4, pp. 35–45, 2013.

[6] K. Bowman, C. Tokunaga, J. Tschanz, and A. Raychowdhury, "Dynamic variation monitor for measuring the impact of voltage droops on microprocessor clock frequency," in *Custom Integrated Circuits Conference*, 2010, pp. 1–4.

[7] N. James, P. Restle, J. Friedrich, and B. Huott, "Comparison of split-versus connected-core supplies in the power6 microprocessor," in *IEEE International Solid-State Circuits Conference*, 2007, pp. 298–604.

[8] X. Liu, S. Sun, X. Li, H. Qian, and P. Zhou, "Machine learning for noise sensor placement and full-chip voltage emergency detection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 3, pp. 421–434, 2017.

[9] X. Liu, S. Sun, P. Zhou, and X. Li, "A statistical methodology for noise sensor placement and full-chip voltage map generation," in *Design Automation Conference*, 2015, pp. 1–6.

[10] V. J. Reddi, M. S. Gupta, G. Holloway, and G. Y. Wei, "Voltage emergency prediction: Using signatures to reduce operating margins," in *IEEE International Symposium on High Performance Computer Architecture*, 2009, pp. 18–29.

[11] T. Wang, C. Zhang, J. Xiong, and Y. Shi, "Eagle-eye: A near-optimal statistical framework for noise sensor placement," in *IEEE/ACM International Conference on Computer-Aided Design*, 2013, pp. 437–443.

[12] Comparison of a simple neural network and a deep learning network, http://www.global-engage.com/life-science/deep-learning-in-digital-pathology/.

[13] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[15] D. Cireşan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," *arXiv preprint arXiv:1202.2745*, 2012.

[16] T. Sakurai and A. R. Newton, "Alpha-power law mosfet model and its applications to cmos inverter delay and other formulas," *IEEE Journal of solid-state circuits*, vol. 25, no. 2, pp. 584–594, 1990.

[17] "https://github.com/pytorch/pytorch."

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.