ML-INSIGHT: Machine Learning for Inrush Current Prediction and Power Switch Network Improvement

Vikram Gopalakrishnan, Bing-Yue Wu, and Vidya A. Chhabria,
Arizona State University

ABSTRACT

Today's large-scale designs utilize power gating to achieve low power consumption. This strategy involves creating an efficient power switch network that considers both the surge current (inrush) and the time it takes for the domain to wake up (wakeup latency). Optimized design of the power switch network requires an approach that minimizes the inrush current while meeting the wakeup latency specification. However, analyzing the network for inrush is computationally very expensive with large runtimes, particularly for complex networks, making an optimization framework that calls the analysis engine under the hood prohibitively slow. To address this challenge, this paper introduces the use of machine learning (ML) techniques to estimate inrush current. The ML-enabled fast inference for inrush prediction is applied to optimize the power switch network to minimize the inrush current and also meet the wakeup latency constraint. The ML model demonstrates a mean error of 5% compared to SPICE simulations, offering an acceleration of over 50×.

ACM Reference Format:
Vikram Gopalakrishnan, Bing-Yue Wu, and Vidya A. Chhabria,. 2024. ML-INSIGHT: Machine Learning for Inrush Current Prediction and Power Switch Network Improvement. In *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED '24), August 5–7, 2024, Newport Beach, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3665314.3670807

1 INTRODUCTION

Modern semiconductor chips, particularly those in battery-powered devices, prioritize energy efficiency highly. However, as technology advances and process technologies shrink, leakage currents increasingly account for a significant portion of total power usage, adversely affecting the longevity of device batteries [1]. Power gating has emerged as a crucial and standard practice within the industry to counteract leakage power dissipation. It does so by controlling the power supply to different chip blocks/domains, thereby effectively reducing unnecessary power consumption by deactivating idle blocks. The introduction of the power gates, results in a complex power network consisting of three components: the permanent power network (VDD), the power switches (PSW), and the gated/switched power network (VDD_SW). Fig. 1 illustrates the model of this power network. The gate/switch is modeled as a transistor, the enable signal determines whether the gated domain is turned ON or OFF, the gated domain is modeled as a capacitance and a leakage current source.

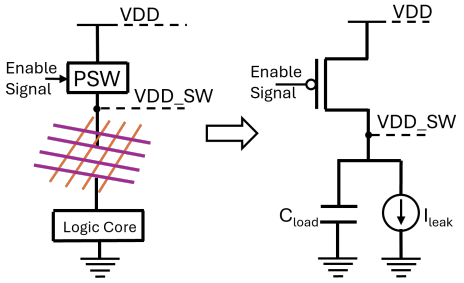

Figure 1: A power switch and its model.

Activating a power-gated block from sleep to active mode causes a significant inrush current due to the simultaneous charging of its internal capacitors which is given by:

$$I_{\text{rush}} = C_{\text{load}} \frac{\partial V}{\partial t} \tag{1}$$

where C_{load} is the capacitance of the domain being gated (shown in Fig 1), and ∂V is the change in voltage during power ramp up and ∂t is the time for power ramp up. This inrush current substantially exceeds the operational current required for the block's active functionality, potentially causing permanent damage to the circuit, increasing power consumption and voltage drops, and, in battery-operated systems, reducing battery life due to the increased load current. The industry's standard practical method to manage inrush current involves using multiple (hundreds of) power switches linked in a hybrid daisy chain [2]. These switch cells are progressively activated in a sequence delayed by a buffer chain, with the inrush current managed by the buffer delay, which sets the sequential activation timing of the switch cells. If the buffer delay is too short, the switch cells turn on almost simultaneously (hammer approach), possibly causing

ISLPED '24, August 5–7, 2024, Newport Beach, CA, USA

ACM ISBN 979-8-4007-0688-2/24/08…$15.00
https://doi.org/10.1145/3665314.3670807

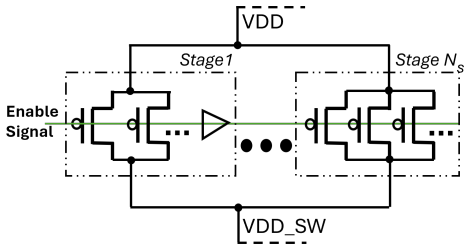

Figure 2: Power switch network (daisy chain topology).

a large rush current. Conversely, a fully daisy approach with one switch per stage results in prolonged wakeup latency due to the slow charging to full voltage. Thus a hybrid daisy chain combines the benefits from both approaches.

Fig. 2 shows the model of a power switch network (PSN) in a daisy chain. Designing this network for a given power domain is extremely challenging as it requires determining the total number of stages, and the number of switches per stage (as shown in the figure) to minimize the inrush current while also satisfying the wakeup latency constraint. This challenge is compounded by the computationally intensive nature of inrush current estimation, requiring circuit simulation using SPICE that solves Equation (1) numerically, making any PSN design optimization framework that incorporates inrush analysis under the hood prohibitively slow.

Our proposed work, ML-INSIGHT – Machine Learning for Inrush Current Prediction and Power Switch Network Improvement, seeks to address the scalability issue. ML-INSIGHT develops a machine learning (ML) model (multilayer perceptron) to rapidly predict peak inrush current using the PSN pattern, capacitance (C_{load}), and leakage current (I_{leak}) as features. The model once trained can be transferred across different designs, patterns, and different numbers of switches. ML-INSIGHT then uses the model in an exploration framework to identify the PSN topology that results in the least inrush current while meeting the wakeup latency constraints. This tool can determine the best PSN layout before floorplanning, which can then be implemented with any commercial EDA tool. Our key contributions are as follows:

(1) To the best of our knowledge, this is the first work to apply ML to address challenges in PSN optimization.
(2) We develop an ML model to predict peak inrush current and leverage the model for PSN design space exploration.
(3) ML-INSIGHT can predict peak inrush with an average accuracy of 95% and a speedup of over 50× compared to SPICE. The model is transferable across designs (C_{load} and I_{leak}) and a different number of power switches.
(4) ML-INSIGHT enables rapid exploration of the PSN patterns with over 50× speedup compared to SPICE-based exploration and an accuracy of over 98%.

ML-INSIGHT is open-source and available at [3].

2 PRELIMINARIES

Prior Work in PSN optimization and inrush current estimation has been in the late 2000's and early 2010's. The past research can be categorized into three classes. The first class aims at finding the optimal placement of power switch in the design in such a way to reduce the total number of switches while meeting the voltage drop, inrush, and wakeup constraints [4, 5]. The second class aims at minimizing the wakeup latency while ensuring the inrush current constraint is met [6–11]. The third class, aims at reducing inrush current for a given latency specification [12–15].

In [12], the inrush current is reduced by applying a charge recycling circuit, which requires change in power switch design. In [13], the authors partition the design into multiple blocks and activate each block in a particular order to minimize inrush current, but partitioning based on PSN requirement is practically hard to implement. The work in [14] utilizes a voltage sensor to detect the switched voltage and determine the activation times of the switches. In [15], a genetic algorithm-driven method is employed to find an optimal sequence to turn on different gated domains in a chip, aiming to decrease the global inrush current.

Although ML-INSIGHT falls into the third class of research, we approach the problem differently. While the previously mentioned works change the power switch design or require additional design components such as voltage sensors, ML-INSIGHT addresses minimizing the inrush current by optimizing the pattern of power switches in the network before the floorplanning stage of physical design. Further, compared to [10] which uses a simple model based on Equation (1) to estimate inrush that can be very inaccurate. ML-INSIGHT leverages an ML model trained from golden SPICE simulations for a faster and more accurate solution.[1]

Problem Formulation In industrial power-gated designs, the power supply network design is done in two modes: the operational mode and the wakeup mode. During operational mode design, with all power switches activated, the permanent (VDD) and gated power networks (VDD_SW) are linked to create a unified power supply system. The goal here is to reduce the area and operational IR drop impact by the inserted power switches while also ensuring the design meets the operational voltage drop constraints. Therefore, the operational mode design determines the number of switches in the PSN. In the wakeup mode design, the switches in the PSN are arranged in a daisy chain style with an optimal power-on sequence pattern that minimizes the inrush current during wakeup period while the meeting wakeup

[1] It's important to note here that there is no direct prior work that solves the inrush analysis and minimization problem as a PSN pattern selection problem defined in Section 2 making apples-to-apples comparisons challenging. Therefore, we perform comparisons against golden SPICE simulations.

latency constraint. This paper explores the challenges for designing power switch networks for wakeup mode.

Therefore, in the wakeup mode design of the PSN, the number of switches, N_T, to be used for a specific domain are known and are pre-determined based on the operational voltage drop requirements and operational mode design. While fewer switches could lower inrush current and contribute a smaller area footprint, it could lead to increased voltage drops during operation. Therefore, the goal of PSN optimization in the wakeup mode is to synthesize a network that uses exactly N_T switches, meets the wakeup latency constraints T_c, and results in the least possible inrush current I_{rush} for a given power domain characterized by I_{leak} and C_{load}, and the unit buffer delay of the technology library T_{buffer}. The problem is defined as follows:

$$\begin{aligned} &\text{Minimize: } I_{\text{rush}} \\ &\text{Subject to: } T_{\text{wakeup}} \leq T_c \end{aligned} \tag{2}$$

where T_{wakeup} is the wakeup latency of the network and its determined by the number of stages in the daisy chain (shown in Fig. 2) and T_{buffer}. One way to solve this problem is to fix the number of stages in the PSN such that the wakeup latency constraint is met then, then work with minimizing the inrush current for the fixed number of stages. Therefore, we identify the maximum number of stages we can have in the daisy chain N_S such that T_{wakeup} is less than T_c. The maximum number of stages is given by $N_S = \left\lceil \frac{T_c}{T_{\text{buffer}}} \right\rceil$ [10, 16][2] Considering the objective is to identify a network structure that minimizes I_{rush}, employing the longest possible daisy chain would achieve the least I_{rush} [17, 18].

Therefore, the problem is now converted into distributing N_T switches across N_S stages to design a network with the least I_{rush}. Although it may seem that the optimization problem is now of limited scope, it is still a practical and extremely challenging problem to solve as the solution space is of size $\binom{N_T-1}{N_S-1}$ and for $N_S = 6$, and $N_T = 100$, the peak inrush current can vary by 30% for a given power domain with $C_{\text{load}} = 1pF$ and $I_{\text{leak}} = 10\mu A$ as shown in Fig. 3.

3 ML-INSIGHT FRAMEWORK

Addressing the optimization problem formulated in Section 2, requires an exploration framework that can search through the $\binom{N_T-1}{N_S-1}$ different configurations of PSNs to identify the pattern that results in the least inrush. The exploration framework must perform inrush current analysis where the input to the analysis tool is the distribution of the N_T switches across the N_S stages (patterns), and the C_{load}, I_{leak} of the power domain. ML-INSIGHT trains an ML model to predict

[2]We have not included wire delays in this equation since we are working before the flooplanning stage. However, this could be modified to include some pessimism that accounts for wire delay based on wire load models.

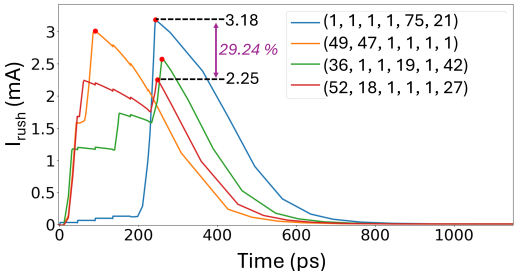

Figure 3: SPICE simulation-based I_{rush} for different patterns PSN for a $C_{\text{load}} = 1pF$ and $I_{\text{leak}} = 10\mu A$.

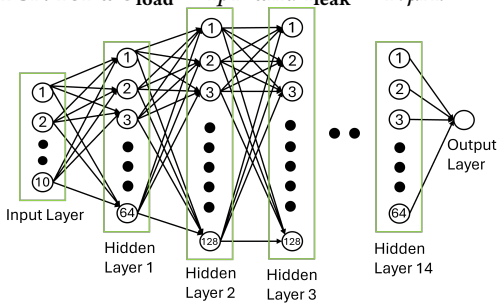

Figure 4: MLP architecture with 14 hidden layers.

the inrush current for a given number of stages N_S. The trained model once trained is transferable across different domains (different C_{load} and I_{leak} values), and various number of switches (different values of N_T), but is specific to N_S.

3.1 Inrush prediction: Feature engineering

ML-INSIGHT aims to predict inrush current by training an ML model. For accurate prediction, it is critical to account for important features that impact inrush.

Feature extraction Given a PSN pattern as in Fig. 2, ML-INSIGHT uses the following features to train the ML model.

(1) Number of switches in each stage: An array of size N_S that lists the number of switches per stage.

(2) Slews of the enable signal at each stage: An array of size N_S that lists the input transition times at each stage. The slew of the enable signal at each stage is estimated based on the strength of the buffer driving the next stage and the number of switches in the next stage. We extract this information from pre-characterized timing libraries.

(3) Leakage (I_{leak}) : We estimate the leakage power of the domain by running power analysis on the netlist.

(4) Load capacitance (C_{load}) : The capacitance of the domain is the sum of the capacitance of the power grid and the capacitance of the instances in the domain connected to the power-gated domain. The power grid capacitance is extracted by multiplying the per-unit capacitance of the metal layers used in the power grid and the length of the power grid wires.

Feature importance To demonstrate that each selected feature is indispensable to the model, we perform a sensitivity analysis by measuring the mean absolute error (MAE), and

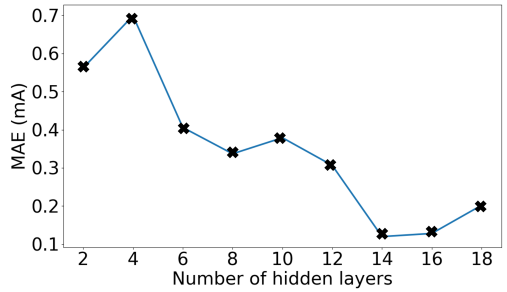

Figure 5: Hyperparameter exploration: Test MAE vs. number of hidden layers.

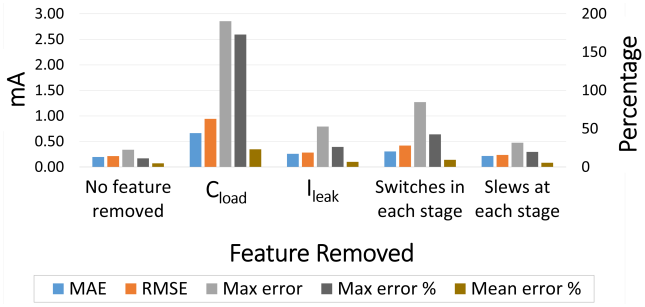

Figure 6: Feature importance and analysis.

root mean square error (RMSE) for different models, each trained by removing one specific feature at a time. The x-axis of Fig. 6 lists the feature that has been removed, and the y-axis highlights the %error (right) and MAE (left) of the testset. The figure shows an error bar that also shows the maximum %error. We find that the model has the best accuracy when all the features are selected.

3.2 Inrush prediction: Model architecture

For inrush prediction, we leverage a multi-layer perception (MLP) as depicted in Fig. 4. The input layer is of size $2\times N_S+2$ corresponding to the slews and number of switches at each stage, and the leakage and capacitance of the gated-domain. The input layer is followed by 14 hidden layers each with 128 nodes except the first and the last hidden layer. The output layer has only one node, representing the maximum inrush current for the PSN topology and the given gated domain. We selected the number of hidden layers by performing hyper-parameter tuning using the synthetic training data. Fig. 5 shows a plot of MAE vs. the number of hidden layers and we observed that the MAE was the least with 14 hidden layers.

3.3 Inrush prediction: Model training

Ground-truth data generation Our ground-truth data is generated using SPICE simulations of the PSN in Fig. 2 implemented in a FinFET 7nm technology node [19]. We conducted simulations over a spectrum of N_T values –100, 150, and 200– with each comprising three stages, yielding 4, 851, 11, 026, and 19, 701 distinct patterns, respectively. In addition, we explored a range of C_{load} from 1pF to 24pF and I_{leak} at 1μA, 10μA, and 20μA, culminating in 0.46M, 1.06M, and 1.91M datapoints for each respective N_T grouping. Consequently, our model is informed by an aggregate of 3.4M datapoints. We designate 70% of this dataset for training purposes and the remaining 30% for testing.

Any datapoint configurations utilized in the assessment of our actual circuits (referenced in Table1) are meticulously excluded from the training dataset. We derive the labels for maximum inrush current directly from SPICE for every datapoint, which are then employed to instruct our MLP as depicted in Fig. 4. Although the process of generating ground-truth data is notably slow, taking on average one minute per five thousand (on a 16-core CPU Intel (R) Xeon Gold 6246R CPU @ 3.4GHz) datapoints, it is a one-time cost for each technology. This one-time cost enables the trained model to be applied (transferred) to diverse PSN configurations and designs, ensuring the rapid and accurate inrush prediction. Furthermore, this one-time cost is amortized across the number of times the trained model can be used in inference across multiple designs, multiple power domains, and multiple times in the design cycle.

MLP training The model is trained with RMSE loss function, an ADAM optimizer, a batch size of 64, and 10 epochs. The training is done using Keras ML library in Python [20] and took 18 minutes for training on NVIDIA RTX 5500 GPU.

3.4 Pattern exploration framework

The goal of the pattern exploration framework in ML-SIGHT is to solve the optimization problem defined in Equation (2). Our exploration framework, shown in Fig. 7, consists of a pattern generator that creates the $\binom{N_T-1}{N_S-1}$ number of patterns, and an inrush analysis engine. The range of estimated inrush current can significantly depend on N_S and N_T (as shown in Fig. 3). The generated patterns are individually simulated for their peak inrush current. The inrush analysis engine in "ground-truth" flow simulates the daisy chain PSN shown in Fig. 2 using SPICE while the ML-INSIGHT flow leverages the trained MLP to perform fast accurate inrush analysis. The flow then selects the pattern that results in the least inrush.

4 ML-INSIGHT EVALUATION

We evaluate the proposed inrush prediction and pattern exploration framework versus SPICE for accuracy and speed.

4.1 Experimental setup and testcases

To evaluate ML-INSIGHT, we synthesize 10 designs from open-cores [21], listed in Table 1 and OpenROAD GitHub repository [22] in ASAP7 technology which vary in their number of instances (2,000 to over 200,000). Next, we perform floorplanning and placement using OpenROAD [23]. We extract the leakage power, and the domain capacitance of

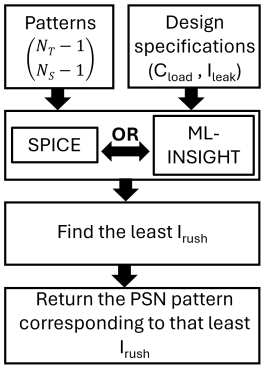

Figure 7: Pattern exploration framework.

Table 1: Circuit testcases in ASAP7 technology node.

Design	N_T	# instances	C_{load} (pF)	I_{leak} (uA)
Ariane	180	93,459	28.653	25000
AES256	175	233,579	22	23.71
Mempool	160	130,460	19.49	3700
Ibex	145	54,785	2.783	3.04
JPEG	140	53,317	6.13	6.57
AES128	140	15,807	2.048	1.89
Ethmac	140	162,614	9.148	11.74
Raven_SHA	135	29,336	3.69	3.71
Mock-array	120	58,666	10.198	0.47
Amber	110	2,747	1	0.68

each of these designs and treat them as individual gated domains. We determine the total number of switches N_T per gated domain (or design) based on voltage drop requirements (5% of VDD). For our testcases, we assume $N_S = 3$ to meet the wakeup latency constraints for these designs in a 7nm technology. We have developed ML-INSIGHT in Python3.9. Our experiments are performed on a 16-core Intel (R) Xeon Gold 6246R CPU @ 3.4GHz with an NVIDIA RTX A5500 GPU and all reported runtimes are on this machine.

4.2 MLP Evaluation

ML-INSIGHT accuracy vs. SPICE Fig. 8 shows a scatter plot demonstrating the accuracy of our MLP in predicting the max inrush current of our testset (unseen during training). This figure shows the accuracy of the 30% of the 3.4M datapoints we generated (Section 3.3). ML-INSIGHT is transferable across a diverse range of N_T. We trained the model on N_T values of 100, 150, and 200 and performed predictions on 130, 160, and 190. The small errors show that the model is transferable across different N_T values.

Fig. 9 shows the accuracy of the model inference on the testcases shown in Table 1. Although, there are 10 testcases listed in Table 1, the figure has several more datapoints and only 8 unique testcases as we predict the inrush current for different possible PSN patterns for each testcase. To ensure that the range of the x and y axes is not too large across

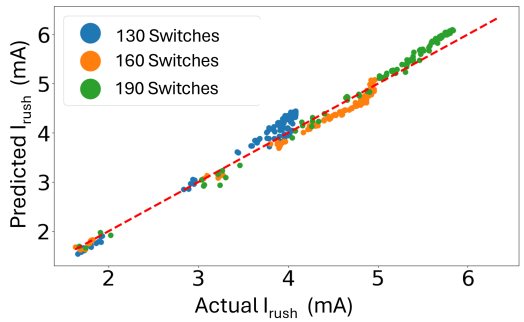

Figure 8: ML-INSIGHT accuracy: Predicted vs. SPICE peak inrush on our synthetic testset for different N_T.

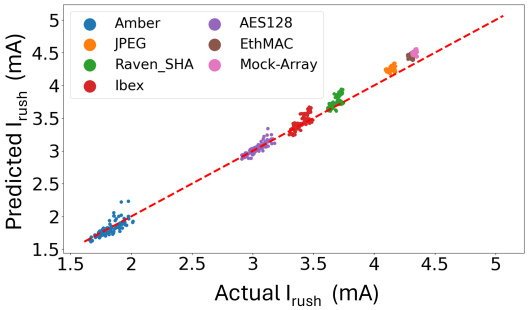

Figure 9: ML-INSIGHT accuracy: Predicted vs. SPICE peak inrush on our synthetic testset with $N_T = 140$.

Table 2: ML-INSIGHT accuracy vs. SPICE for inrush prediction on 100 different patterns.

Design	N_T	MAE (mA)	RMSE (mA)	Max error (mA)	Max % error	Mean % error
Ariane	180	0.353	0.375	0.532	9.243	6.142
AES256	175	0.378	0.388	0.544	9.828	6.814
Mempool	160	0.115	0.138	0.288	5.703	2.273
Ibex	145	0.028	0.036	0.104	2.994	0.819
JPEG	140	0.116	0.120	0.181	4.368	2.815
AES128	140	0.034	0.046	0.218	7.006	1.122
Ethmac	140	0.162	0.164	0.199	4.615	3.775
Raven_SHA	135	0.065	0.088	0.211	5.677	1.784
Mock-array	120	0.140	0.169	0.314	8.383	3.745
Amber	110	0.047	0.068	0.300	16.485	2.673

the diverse testcases, we use a fixed number of switches and show datapoints only a subset (8 of 10) of the testcases. The scatter points lie along the 45-degree line indicating accurate prediction on the real circuit testcases.

Table 2 lists the MAE, mean, and max percentage errors for all testcases described in Table 1. The percentage error is the absolute difference between the predicted and SPICE inrush current divided by the SPICE inrush current. The trained model is applied to all testcases for a fast inrush prediction. The worstcase max error of ML-INSIGHT method is 16.5% and the worstcase mean error is less than 4% showing transferability of the model across different designs and N_T values. The table lists the mean and max error for 100 random PSN patterns showing transferability across patterns.

Table 3: Pattern exploration results from ML-INSIGHT and SPICE with the least I_{rush}.

Design	ML-INSIGHT best pattern found			SPICE best pattern found		Comparison in least I_{rush}		
	Pred. I_{rush} (mA)	SPICE I_{rush} (mA)	Run-time (s)	SPICE I_{rush} (mA)	Run-time (s)	Error I_{rush} (mA)	%Error I_{rush}	Speed-up with parallelism
Ariane	5.721	5.753	0.63	5.744	4,713	0.009	0.16	74.28×
AES256	5.721	5.567	0.47	5.533	4,707	0.034	0.61	99.21×
Mempool	4.728	5.065	0.45	5.056	3,816	0.009	0.18	85.26×
Ibex	3.341	3.453	0.38	3.435	2,859	0.018	0.52	74.33×
JPEG	4.165	4.17	0.33	4.088	2,846	0.082	2.00	86.14×
AES128	2.86	2.918	0.34	2.898	2,672	0.02	0.69	78.77×
Ethmac	4.315	4.315	0.31	4.267	2,654	0.048	1.12	85.69×
Rav.SHA	3.596	3.639	0.31	3.608	2,634	0.031	0.86	85.68×
MockArr.	3.655	3.759	0.26	3.731	1,932	0.028	0.75	74.22×
Amber	1.590	1.658	0.34	1.639	1,709	0.019	1.16	50.81×

4.3 Exploration framework evaluation

Accuracy vs. a SPICE-based exploration Table 3 lists the SPICE and ML-INSIGHT inrush currents for the ML-INSIGHT-generated pattern which gives minimum inrush current. Similarly, the pattern corresponding to the least inrush current is found from the SPICE and its SPICE-generated I_{rush} values are listed. The last two columns in the table compare SPICE and ML-INSIGHT. The columns list the absolute and percentage error in the least inrush current reported by ML-INSIGHT. It shows that the difference in the minimum inrush between ML-INSIGHT and SPICE is very small (less than 0.082mA, i.e., less than 2%). Considering that the peak inrush current can fluctuate by up to 30% for a power domain by varying the PSN pattern (Fig. 3), an error of 2% is insignificant. This shows that the trained model is transferable across different designs and can result in patterns that have inrush currents close to the SPICE least inrush.

4.4 Runtime comparison

Table 3 shows the speedup ML-INSIGHT provides, considering parallel SPICE simulations. Given our access to 100 SPICE licenses and the capability to execute these simulations concurrently, our minimum speedup exceeds 50×. The runtimes listed in seconds are assuming we run SPICE sequentially for all the $\binom{N_T-1}{N_S-1}$ patterns. However, the speedup column accounts for the parallel SPICE simulations. This shows the scalability of ML-INSIGHT, which is crucial for large N_T and N_S values. The speedup comes at a very small error in the least inrush current (less than 2%) and is license-free.[3]

5 CONCLUSION

This paper presents ML-INSIGHT, an ML-based inrush current predictor. It is transferable across different designs but is specific to a given wakeup latency and technology node. ML-INSIGHT enables fast PSN pattern exploration to minimize inrush, meeting latency constraints with <2% error and over 50x faster than SPICE. ML-INSIGHT is available at [3].

[3]Runtime for feature extraction is common to both SPICE and ML-INSIGHT.

REFERENCES

[1] V. Sreekumar and S. Ravichandran, "Impact of leakage and short circuit current in rush current analysis of power gated domains," in *Proceedings of the IEEE SoutheastCon*, pp. 41–44, 2010.

[2] K. Shi, Z. Lin, and Y.-M. Jiang, "A power network synthesis method for industrial power gating designs," in *Proc. ISQED*, pp. 362–367, 2007.

[3] "PowerSwitch-ML-INSIGHT," 2024. https://github.com/ASU-VDA-Lab/PowerSwitch-ML-INSIGHT.

[4] R. Vilangudipitchai and P. Balsara, "Power switch network design for MTCMOS," in *Proc. VLSI Des. (VLSID)*, pp. 836–839, 2005.

[5] J. Kozhaya and L. Bakir, "An electrically robust method for placing power gating switches in voltage islands," in *Proc. CICC*, pp. 321–324, 2004.

[6] A. Abdollahi, F. Fallah, and M. Pedram, "A robust power gating structure and power mode transition strategy for MTCMOS design," *IEEE T. VLSI Syst*, vol. 15, no. 1, pp. 80–89, 2007.

[7] C.-Y. Chang, P.-C. Tso, C.-H. Huang, and P.-H. Yang, "A fast wake-up power gating technique with inducing a balanced rush current," in *Proc. ISCAS*, pp. 3086–3089, 2012.

[8] Y.-T. Chen, D.-C. Juan, M.-C. Lee, and S.-C. Chang, "An efficient wake-up schedule during power mode transition considering spurious glitches phenomenon," in *Proc. ICCAD*, pp. 779–782, 2007.

[9] S. Kim, S. Paik, S. Kang, and Y. Shin, "Wakeup scheduling and its buffered tree synthesis for power gating circuits," *Integr. VLSI J.*, vol. 53, no. C, p. 157–170, 2016.

[10] S.-H. Chen, Y.-L. Lin, and M. C.-T. Chao, "Power-up sequence control for MTCMOS designs," *IEEE T. VLSI Syst*, vol. 21, no. 3, pp. 413–423, 2013.

[11] A. Ramalingam, A. Devgan, and D. Z. Pan, "Wakeup scheduling in mtcmos circuits using successive relaxation to minimize ground bounce," *J. Low Power Electron.*, vol. 3, pp. 28–35, 2007.

[12] E. Pakbaznia, F. Fallah, and M. Pedram, "Charge recycling in MTCMOS circuits: concept and analysis," in *Proc. DAC*, pp. 97–102, 2006.

[13] A. Davoodi and A. Srivastava, "Wake-up protocols for controlling current surges in MTCMOS-based technology," in *Proc. ASP-DAC*, vol. 2, pp. 868–871 Vol. 2, 2005.

[14] M.-C. Lee, Y.-T. Chen, Y.-T. Cheng, and S.-C. Chang, "An efficient wakeup scheduling considering resource constraint for sensor-based power gating designs," in *Proc. ICCAD*, pp. 457–460, 2009.

[15] H. Jiang and M. Marek-Sadowska, "Power gating scheduling for power/ground noise reduction," in *Proc. DAC*, pp. 980–985, 2008.

[16] V. Singhal, A. Dey, S. Mallala, and S. Paul, "A methodology for early and accurate analysis of inrush and latency tradeoffs during power-domain wakeup," in *VLSI Design and Test*, pp. 294–303, Springer Berlin Heidelberg, 2013.

[17] K. Choi and J. Frenkil, "An analysis methodology for dynamic power gating," *Sequence Design Inc*, pp. 1–13, 2007.

[18] Y.-T. Shyu, J.-M. Lin, C.-C. Lin, C.-P. Huang, and S.-J. Chang, "An efficient and effective methodology to control turn-on sequence of power switches for power gating designs," *IEEE T. Comput. Aid. D.*, vol. 35, no. 10, pp. 1730–1743, 2016.

[19] V. Vashishtha, M. Vangala, and L. T. Clark, "ASAP7 predictive design kit development and cell design technology co-optimization: Invited paper," in *Proc. ICCAD*, pp. 992–998, 2017.

[20] F. Chollet *et al.*, "Keras," 2015. https://github.com/fchollet/keras.

[21] "Open Cores," 2024. https://opencores.org/projects.

[22] "OpenROAD," 2022. https://github.com/The-OpenROAD-Project/OpenROAD.

[23] "OpenROAD-flow-scripts," 2022. https://github.com/The-OpenROAD-Project/OpenROAD-flow-scripts.