Robust Power Estimation and Simultaneous Switching Noise Prediction Methods Using Machine Learning

March 20th, 2019

Robust Simultaneous Switching Noise Prediction for Test using Deep Neural Network

Seyed Nima Mozaffari, Bonita Bhaskaran, Kaushik Narayanun
Ayub Abdollahian, Vinod Pagalone, Shantanu Sarangi

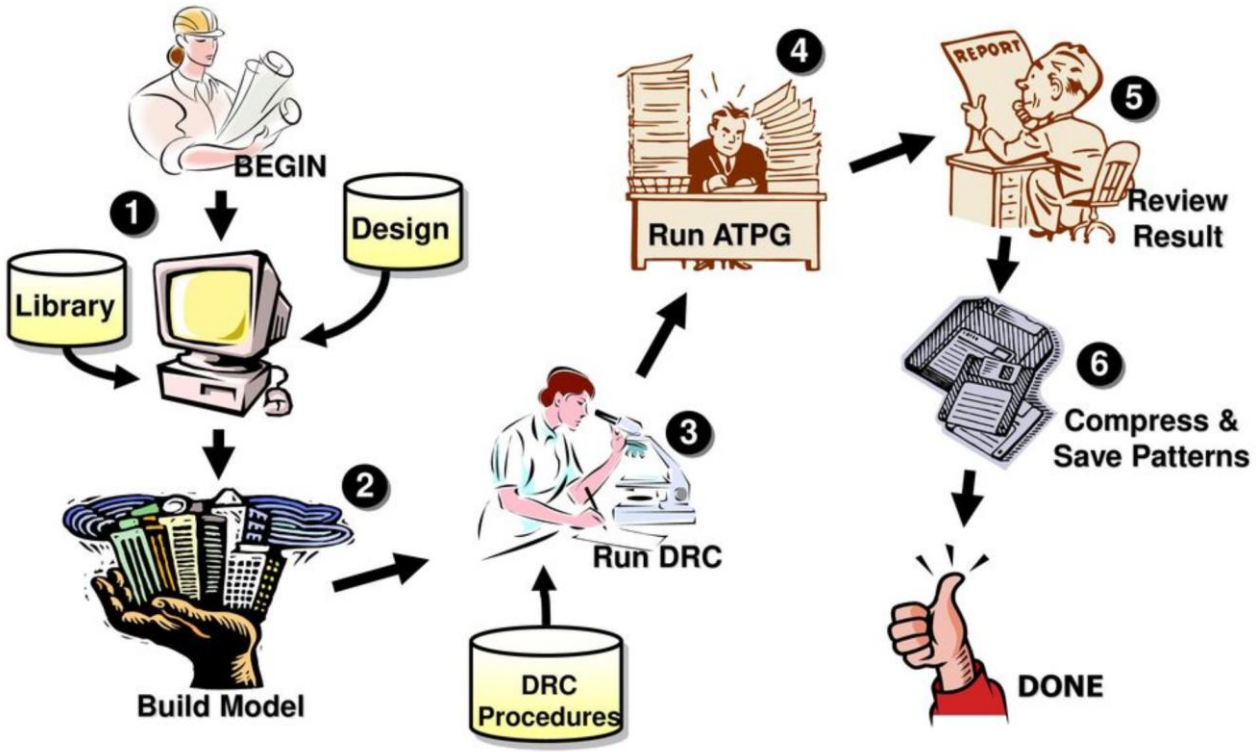
RTL-Level Power Estimation Using Machine Learning

Mark Ren, Yan Zhang, Ben Keller, Brucek Khailany
Yuan Zhou, Zhiru Zhang

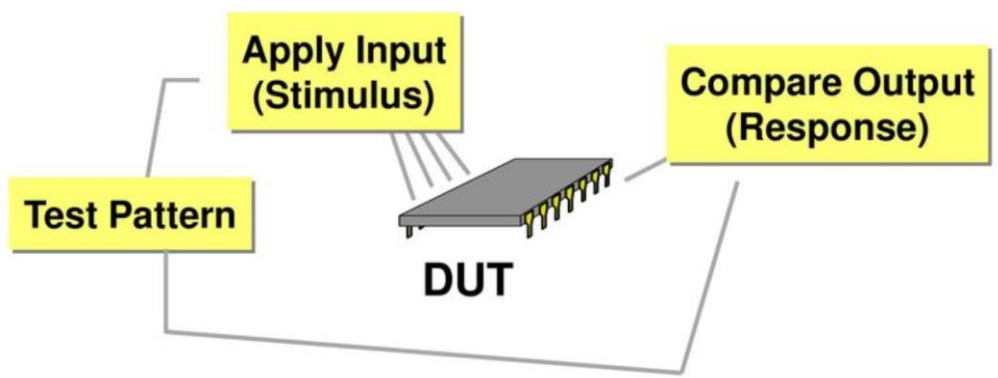
Robust Simultaneous Switching Noise Prediction for Test using Deep Neural Network

Seyed Nima Mozaffari, Bonita Bhaskaran, Kaushik Narayanun
Ayub Abdollahian, Vinod Pagalone, Shantanu Sarangi

DFT – A BIRD'S EYE VIEW

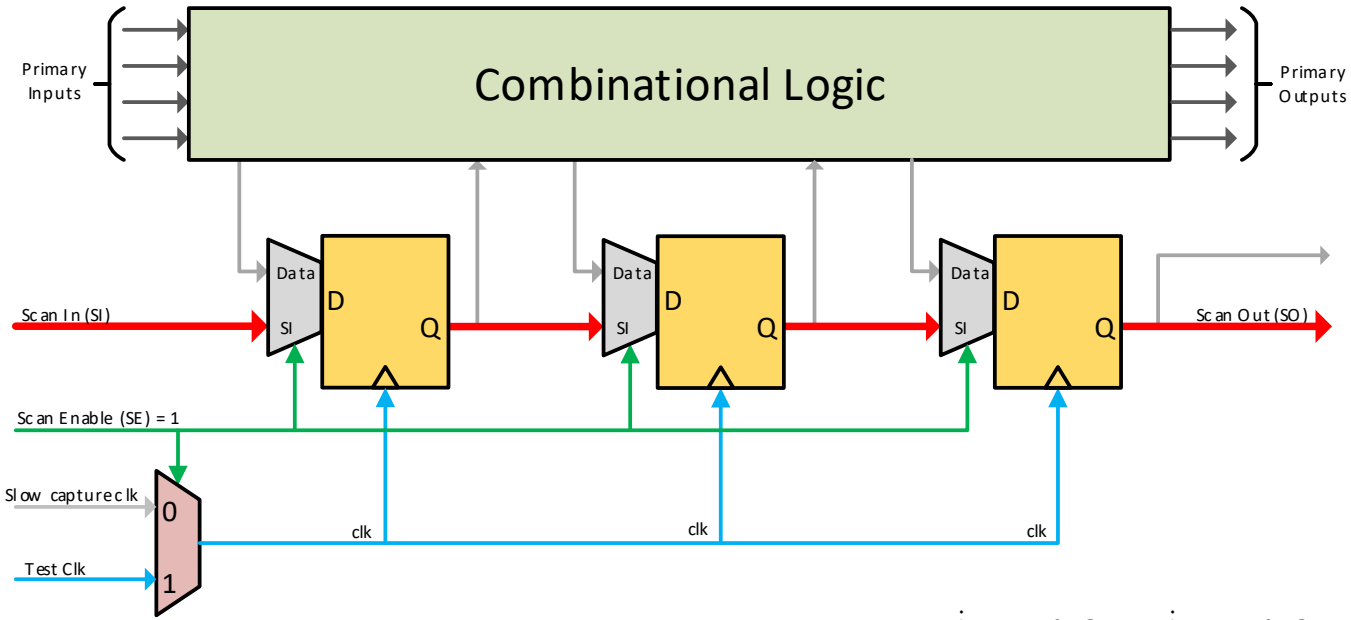


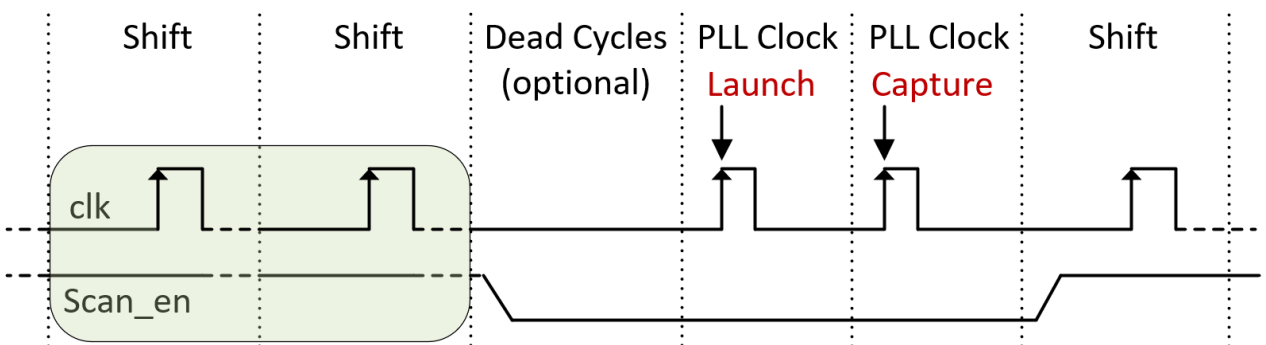
- At-Speed Tests – verify performance
- Stuck-at Tests – detect logical faults
- Parametric Tests – verify AC/DC parameters
- Leakage Tests – catch defects that cause high leakage



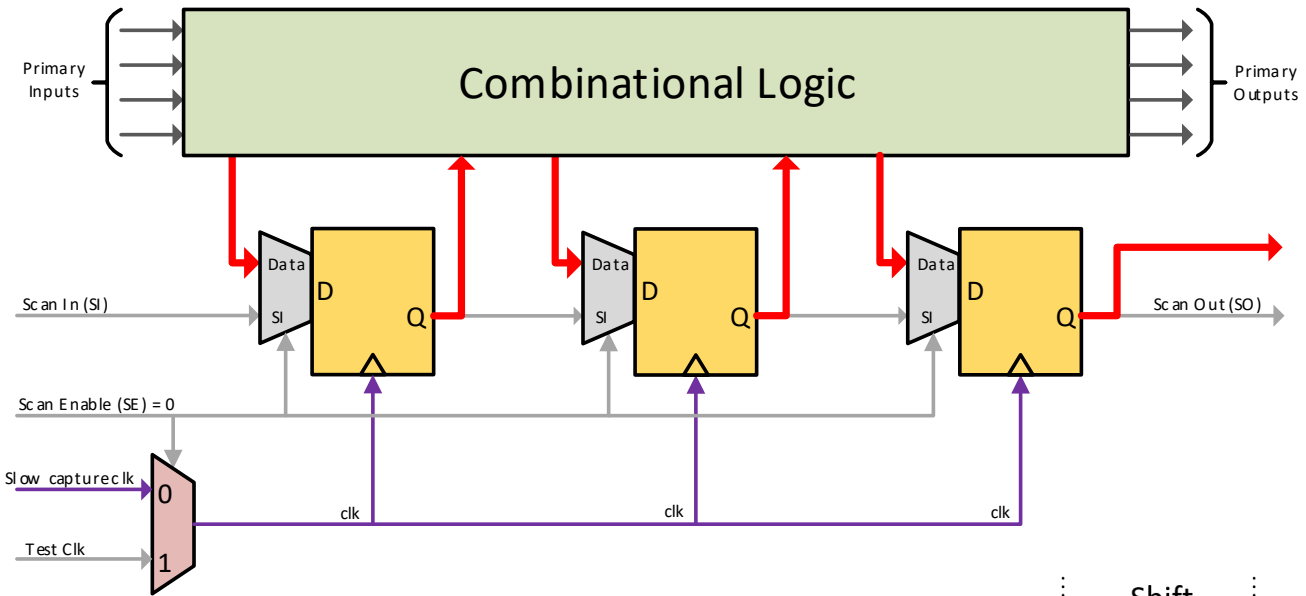
Images – National Applied Research Laboratories

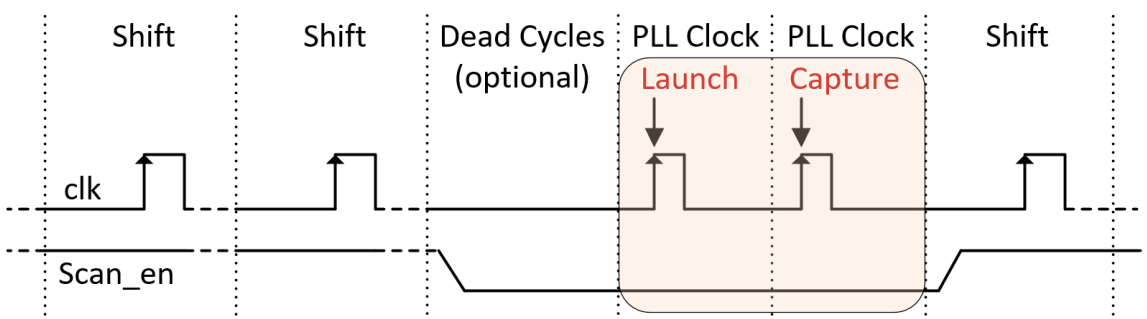
SCAN TEST - SHIFT



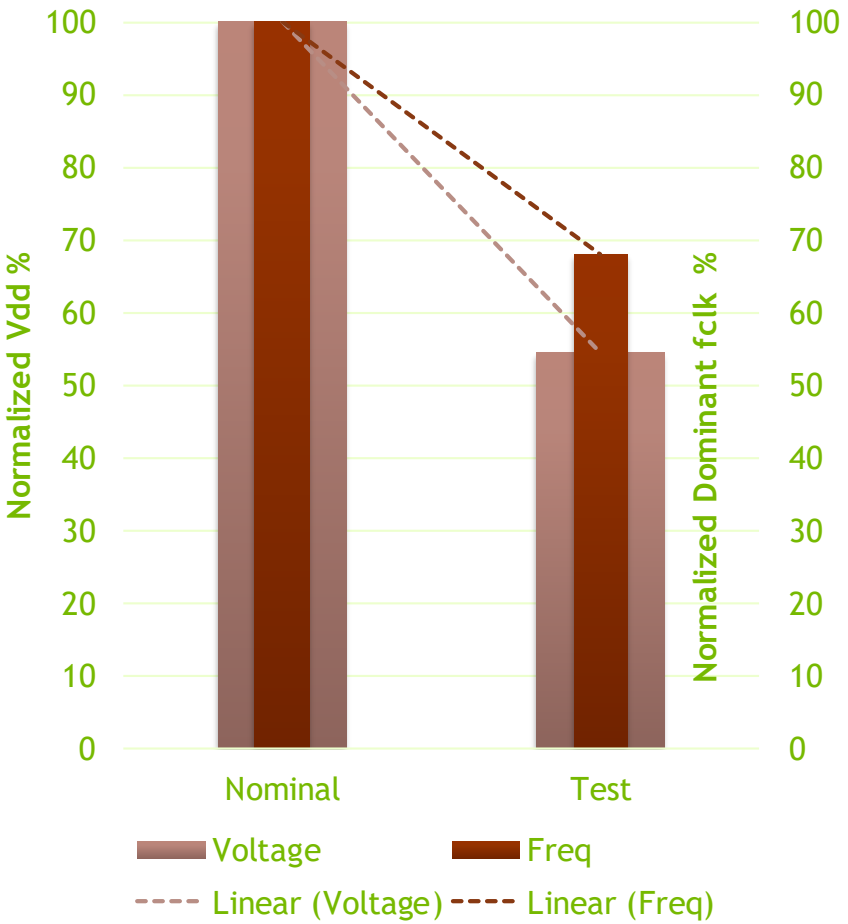


SCAN TEST - CAPTURE





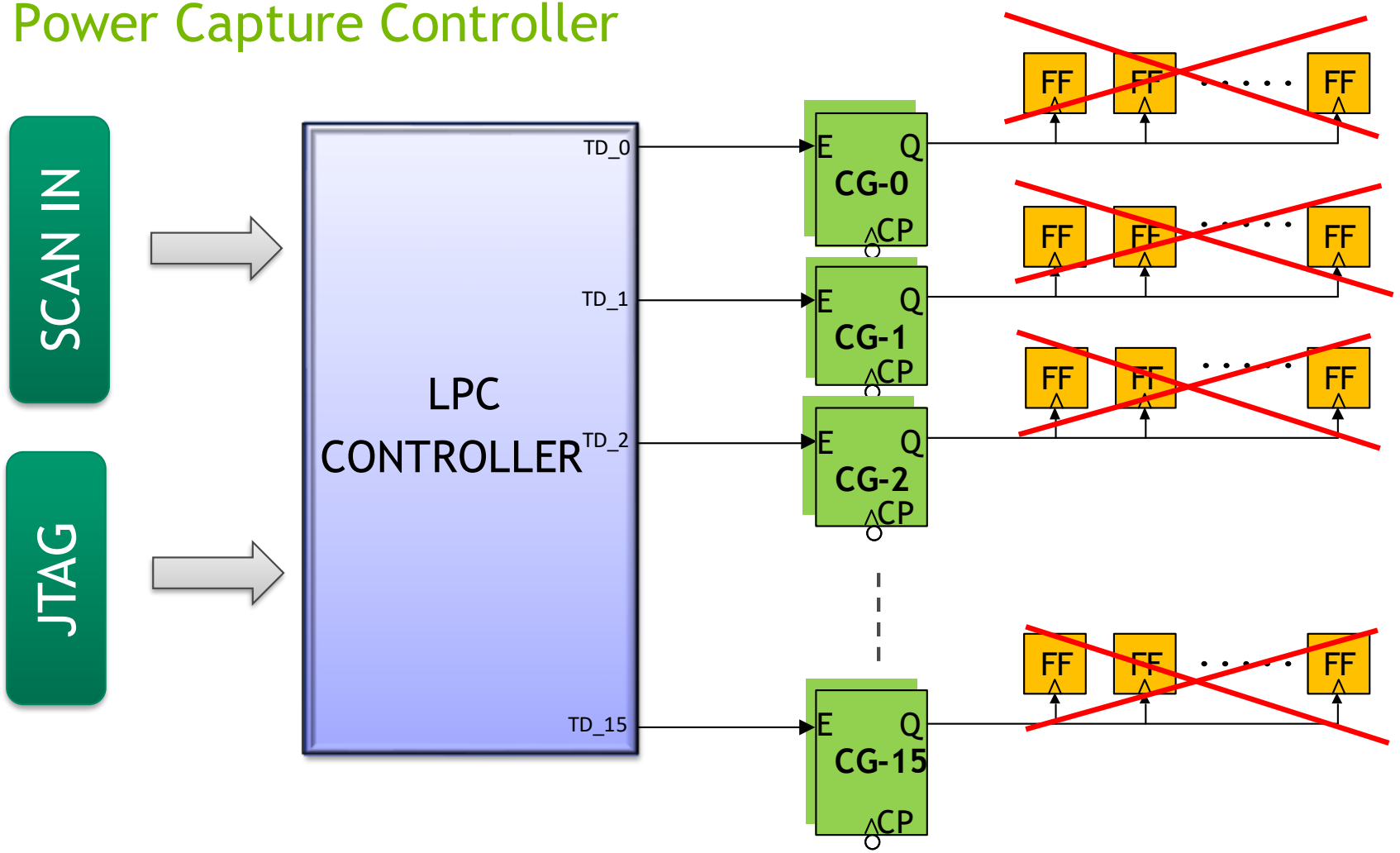
TEST WASTE FROM POWER NOISE



- Power balls overheated; Scan Freq target was lowered
  - Slower frequency → Test Cost
- Higher Vmin issue
  - Vmin thresholds had to be raised; impacts DPPM.
- During MBIST, overheating was observed
  - Serialized tests; increase in Test Time & Test Cost
- Vmin issues observed and being debugged

CAPTURE NOISE

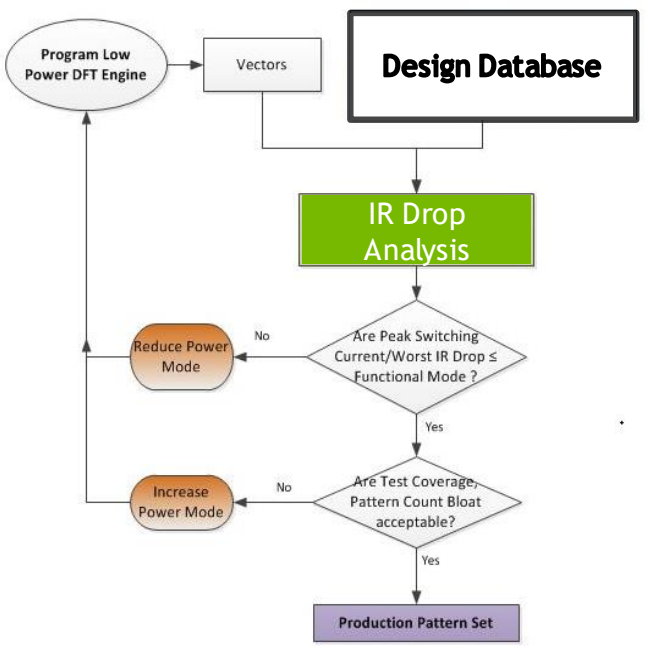
Low Power Capture Controller



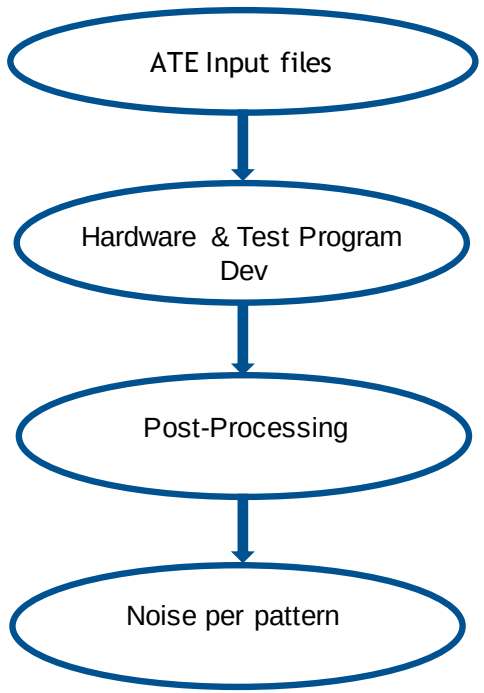
TEST NOISE ESTIMATION

The traditional way

Pre-Silicon Estimation



Post-Silicon Validation



Issues

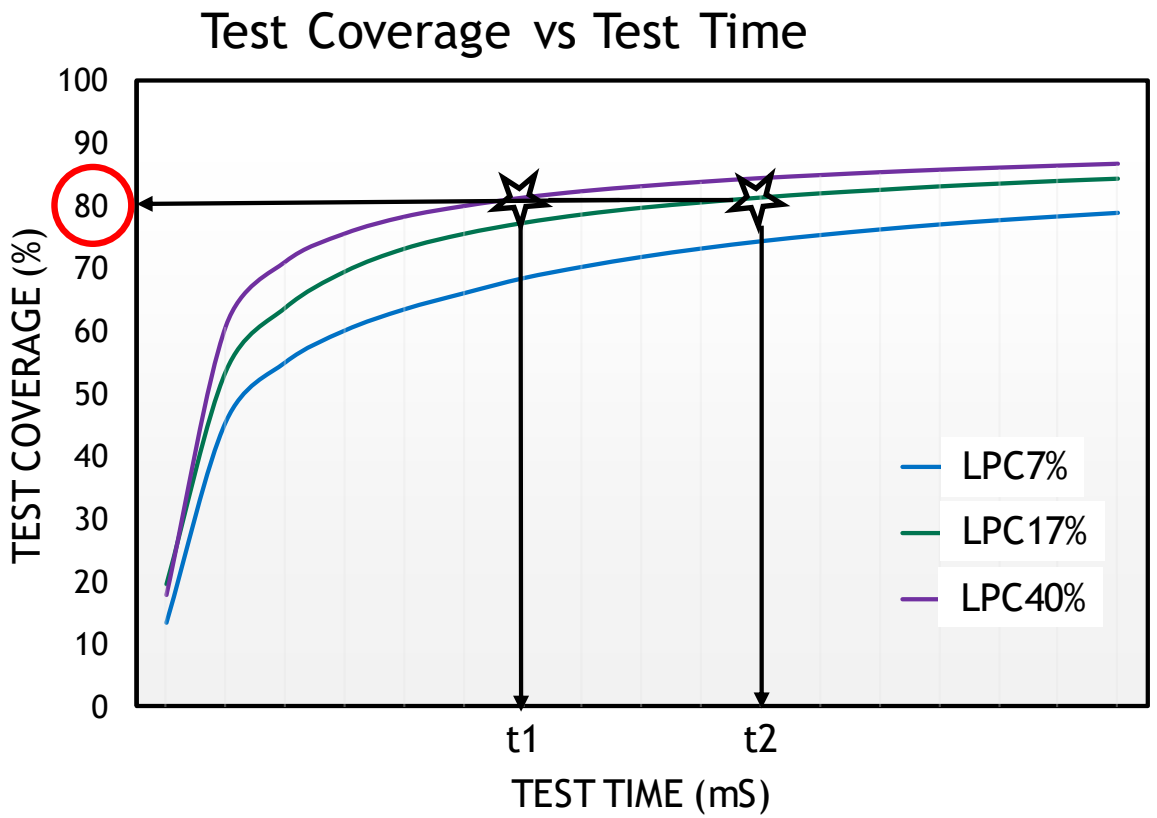
- Can simulate only a handful of vectors
- Not easy to pick top IR-Drop inducing test patterns always
- Machine Time to simulate 3000 patterns is 6-7 years!
- Measurement is feasible for 3-5K patterns

Power noise during test <= functional budget directly impacts test quality !

IMPORTANCE

Strategy – we pick conservative LPC settings!

- Higher Test Time → Higher Test Cost
- For example - Test Time savings of 40% could have been achieved.



Why is Deep Learning a good fit?

- Labeled data is available
- Precision is not the focus
- Need a prediction scheme that encompasses the entire production set

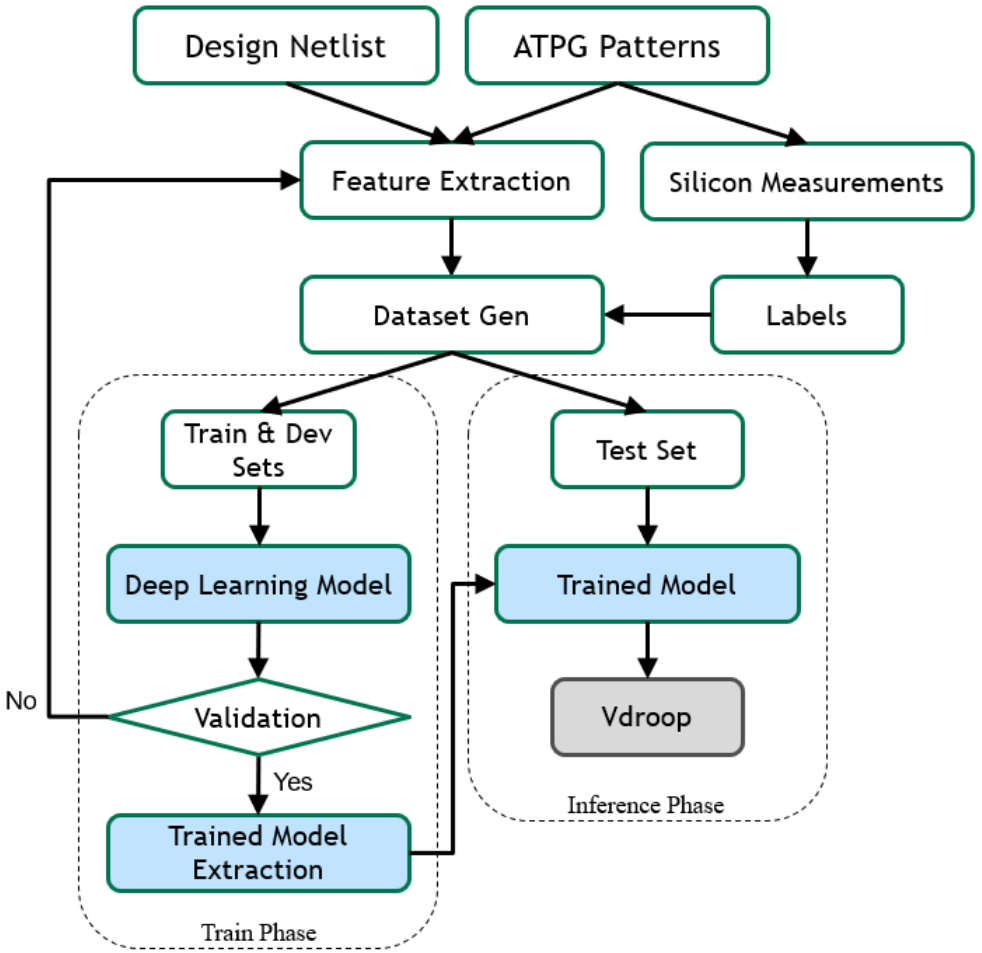
PROPOSED APPROACH

- Design Flow
- Feature Engineering
- Deep Learning Models
- Classification and Regression

PROPOSED APPROACH

- Design Flow
- Feature Engineering
- Deep Learning Models
- Classification and Regression

DESIGN FLOW



Goal:

- Supervised learning model to reduce the time and effort spent
- Most effective set of input features

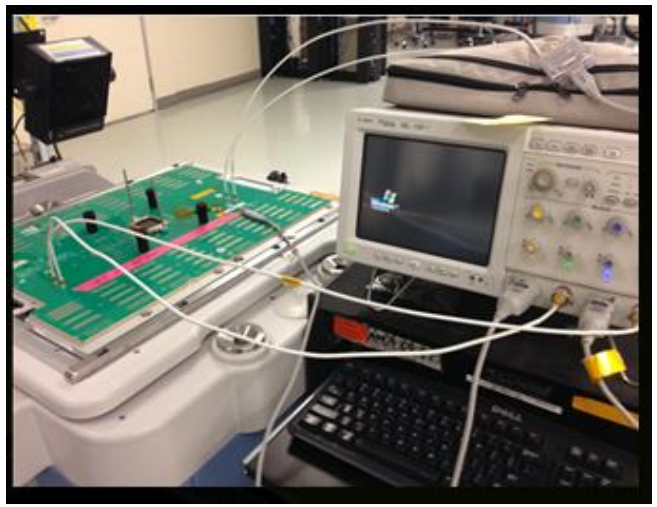
Dataset:

- Input features → parameters that impact the V_{droop}
- Lebels → V_{droop} values from silicon measurements
- Train phase → train:80% & dev:10%
- Inference phase → test:10%

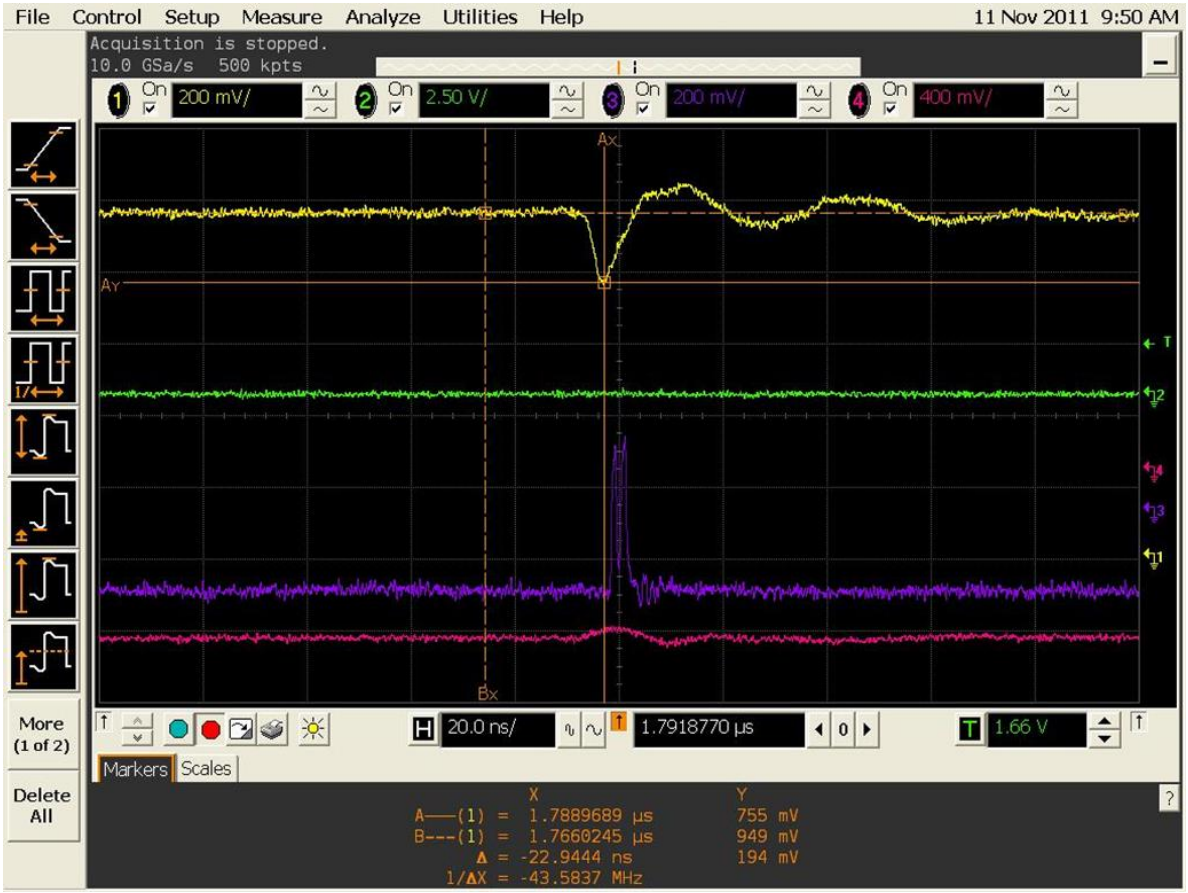
Addresses the following:

- Takes into account all the corner cases for PVT*f* variations
- Helps predict achievable V_{min}
- Cuts down post-silicon measurements – typically 6-8 weeks of engineering effort

HARDWARE SET-UP AND SCOPESHOT

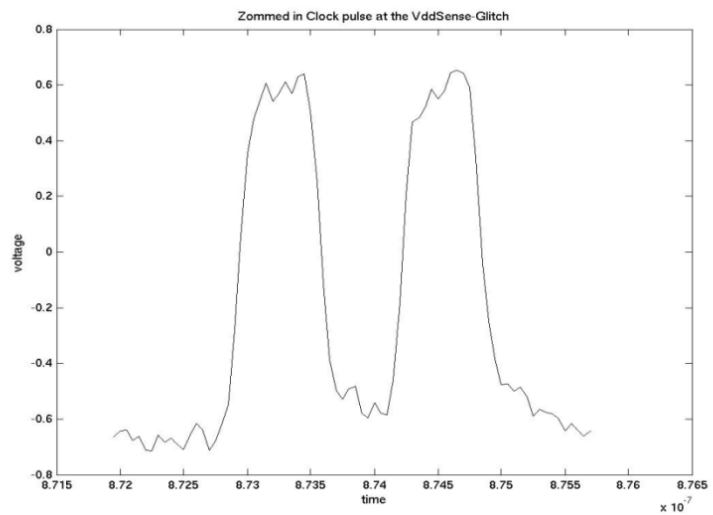


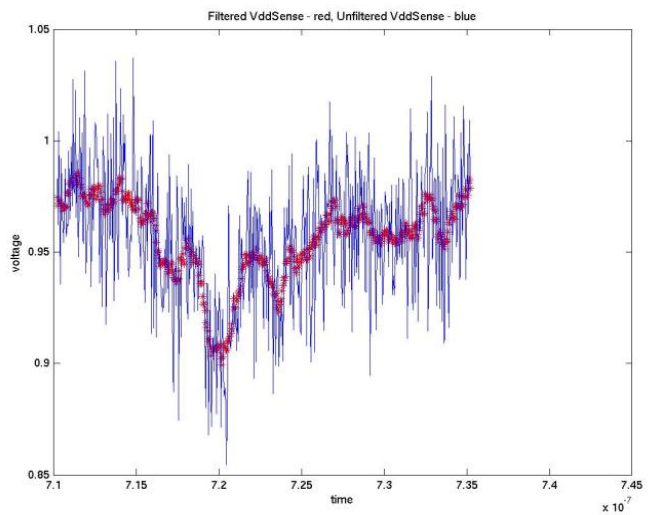
- Yellow – PSN
- Green – Scan Enable
- Purple – CLK
- Pink – Trigger



MATLAB POST PROCESSING

- To be able to accurately tabulate the VDD_Sense droop vs. respective clock domain frequency, a Matlab script is used.
  - Inputs to this script are the stored ".bin" files from the scope
  - Outputs from Matlab script are:

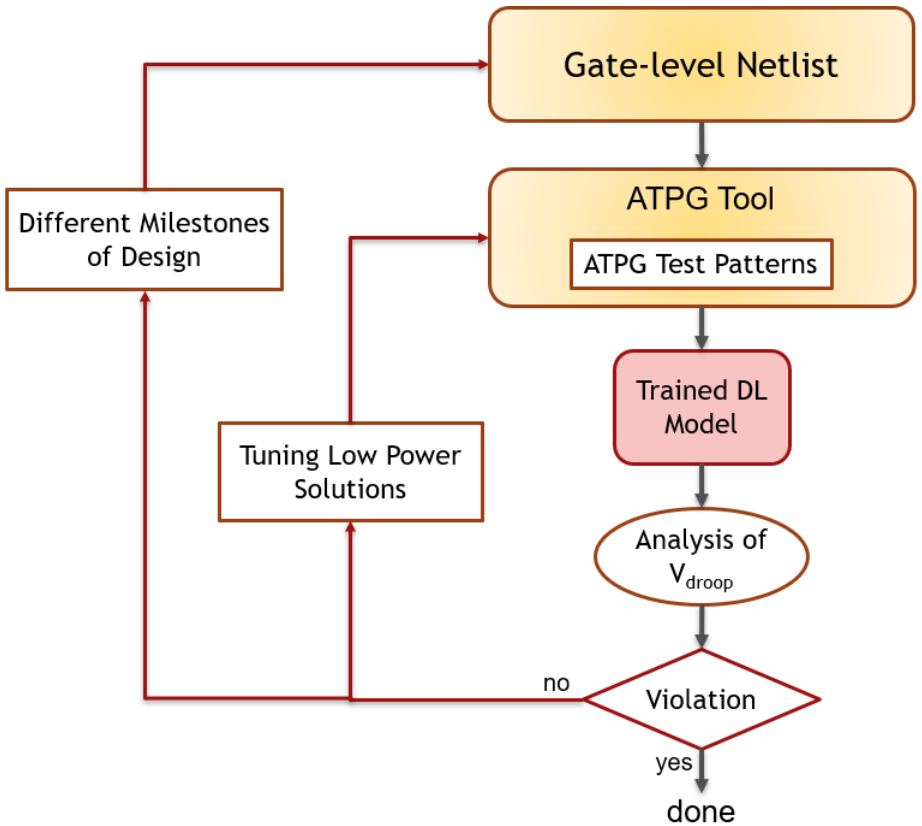




SNAPSHOT OF DATASET

Pattern	Global Switch Factor %	Process	Voltage	Temp	Freq (MHz)	IP Name	Product	LPC	Droop (mV)	Granular Features
1	2.00%	3	1	10	1000	1	2	3	30	
2	3.00%	3	1	10	1000	1	2	3	35	
3	3.00%	3	1	10	1000	1	2	3	35	
4	4.00%	3	1	10	1000	1	2	3	35	
5	3.00%	3	1	10	1000	1	2	3	33	
6	2.00%	3	1	10	1000	1	2	3	33	
7	60.00%	3	1	10	1000	1	2	3	100	
8	45.00%	3	1	10	1000	1	2	3	85	
9	65.00%	3	1	10	1000	1	2	3	105	
10	36.10%	3	1	10	1000	1	2	3	60	
11	36.00%	3	1	10	1000	1	2	3	61	
12	33.00%	3	1	10	1000	1	2	3	60	
13	50.00%	3	1	10	1000	1	2	3	90	
.	.								.	
.	.								.	
.	.								.	
2998	29.87%	3	1	10	1000	1	2	3	55	
2999	47.84%	3	1	10	1000	1	2	3	85	
3000	58.92%	3	1	10	1000	1	2	3	91	

DEPLOYMENT



Goal

- Optimize low power DFT architecture
- Generate reliable test patterns

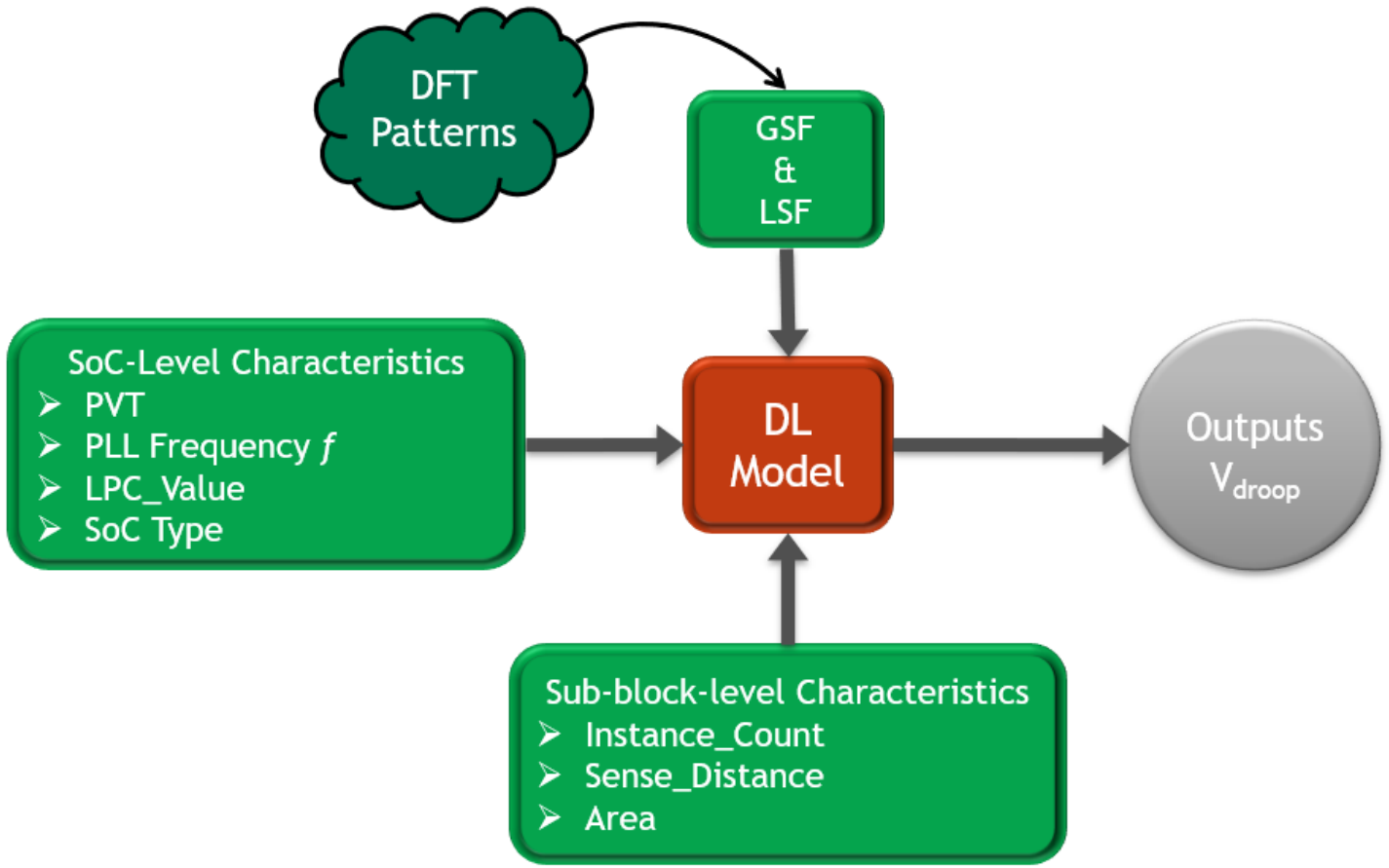
PSN analysis is repeated

- at various milestones of the chip design cycle and finalized close to tape-out.
- until there are no violations for any of the test patterns.

PROPOSED APPROACH

- Design Flow
- Feature Engineering
- Deep Learning Models
- Classification and Regression

FEATURE ENGINEERING



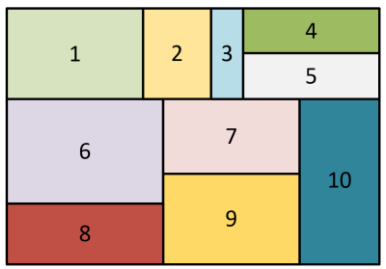
IP-level (Global)

- GSF
- PVT
- PLL frequency *f*
- LP_Value
- Type

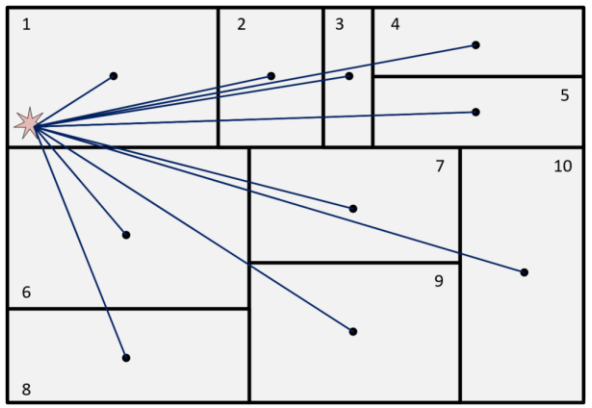
SoC sub-block-level (Local)

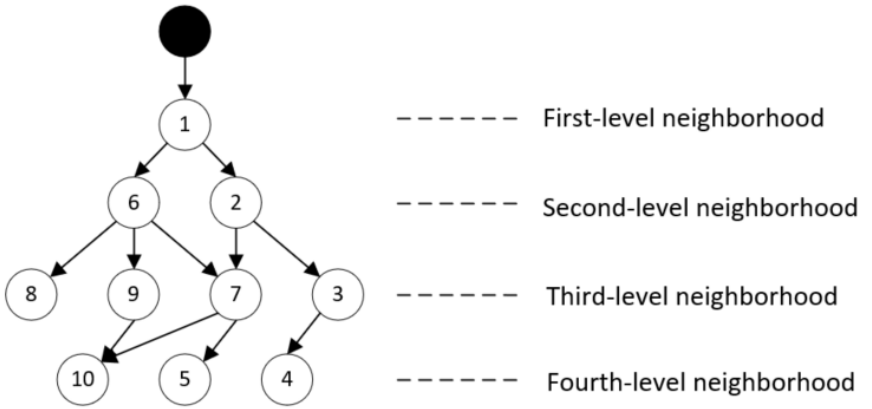
- LSF
- Instance_Count
- Sense_Distance
- Area

EXAMPLE: FEATURE EXTRACTION



Sub-Block-Level layout of an SoC





Global Vector:

36	2	1.0	20	1.0	3	40
GSF (%)	P	V (V)	T (°c)	f (GHz)	Type	LP_Value (%)

- on-chip measurement point location
- sense point neighborhood-level graph
- global and local feature vectors

Local Vector:

0	4	0	1.5	6	4	0	0	4.5	8	10	5	12	2	9	10	9	5	5	11	1.5	5.0	3.0	6.7	6.8	7.5	5.0	9.3	9.0	10	10	5	14	2.5	9	10	9	6.5	6.5	10
LSF (×1000)										Instance_Count (×1000)										Sense_Distance (mm)										Area (mm^2)									

PROPOSED APPROACH

- Design Flow
- Feature Engineering
- Deep Learning Models
- Classification and Regression

DEEP LEARNING MODELS

Fully Connected (FC) model

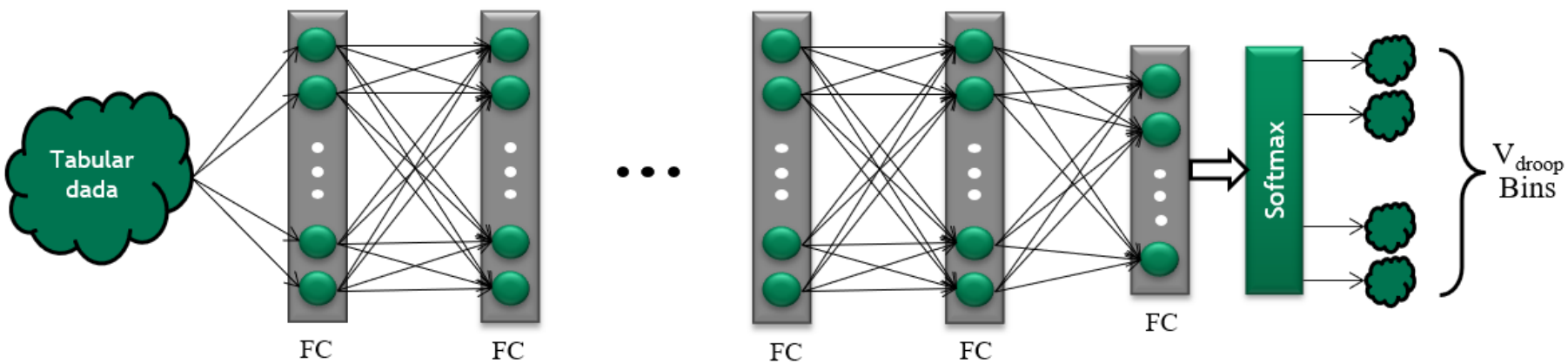
- basic type of neural network and is used in most of the models.
- Flattened FC model
- Hybrid FC model

Natural Language Processing-based (NLP) model

- NLP is traditionally used to analyze human language data.
- we apply the concept of the averaging layer to our IR drop prediction problem.
- Model is independent of the number of sub-blocks in a chip.

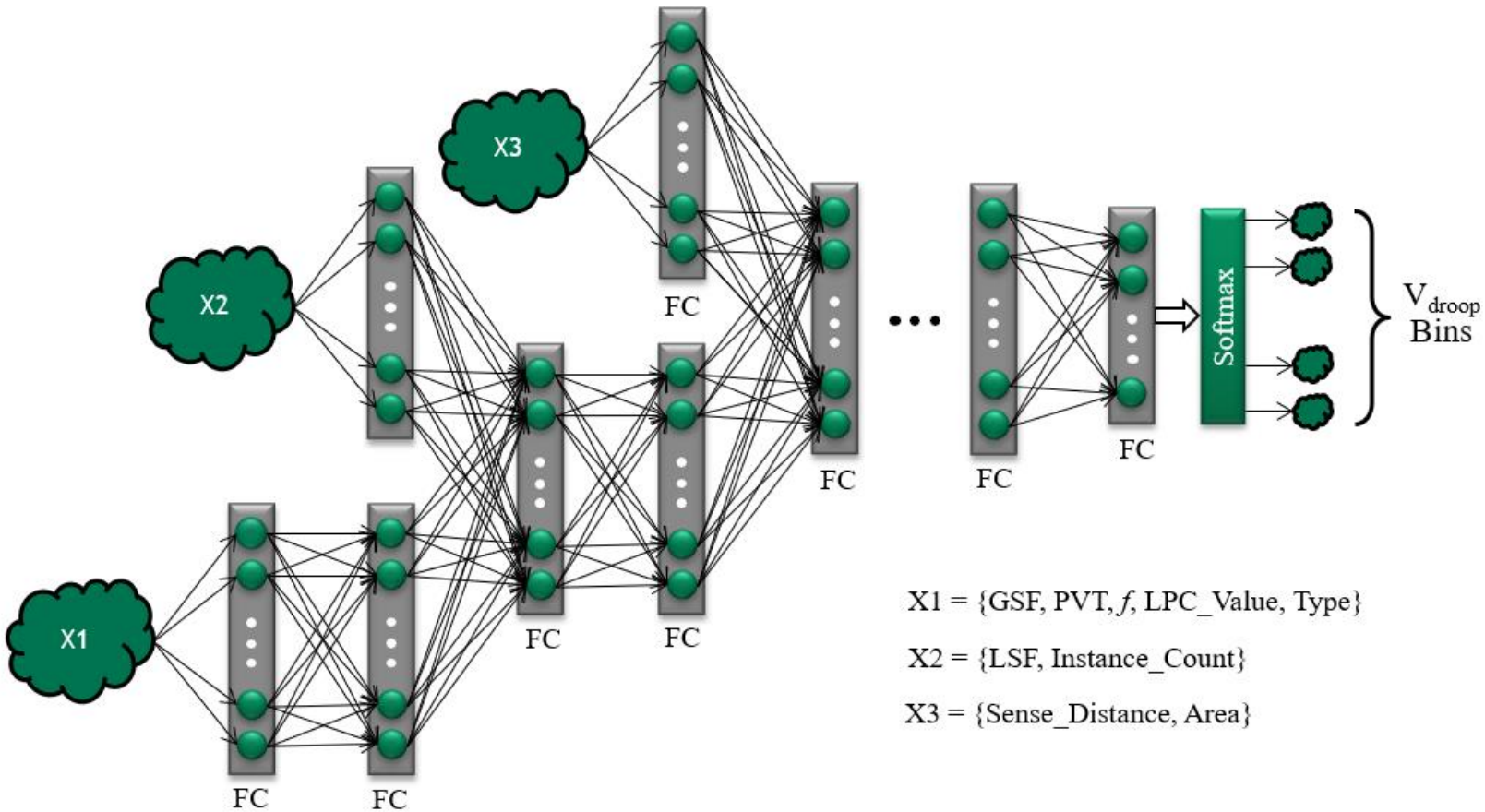
FLATTENED FC MODEL

All the input features are applied simultaneously to the first layer.



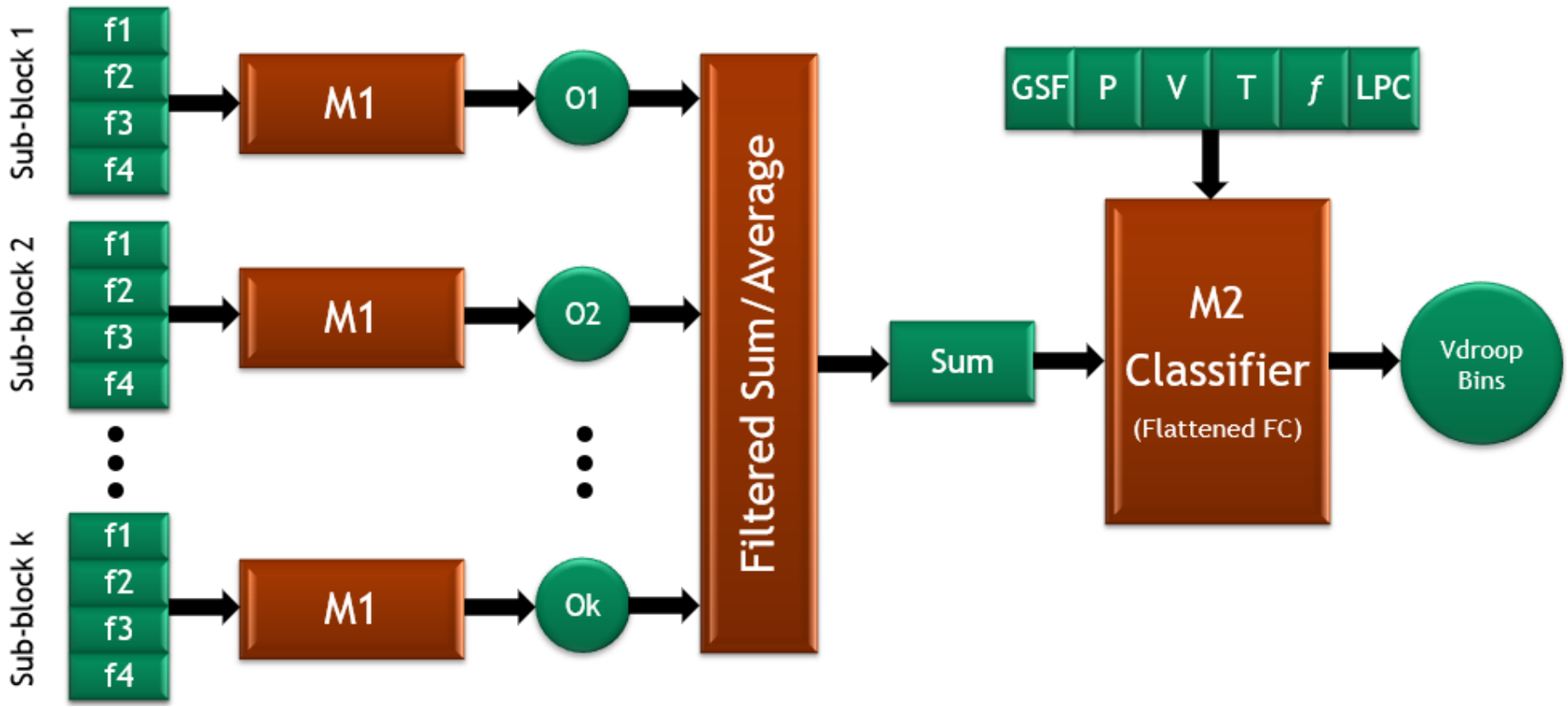
HYBRID FC MODEL

Input features are divided into different groups, each applied to a different layer.



NLP MODEL

- Local features of each sub-block form an individual bag of numbers.
- Filtered Average (FA): 1) filters out non-toggled sub-blocks, 2) calculates the average.



PROPOSED APPROACH

- Design Flow
- Feature Engineering
- Deep Learning Models
- Classification and Regression

CLASSIFICATION AND REGRESSION

- Classification models predict a discrete value (or a bin).
- Regression models predict the absolute value.
- Optimization:

Input Normalization, Adam optimizer, learning rate decay, L2 regularization

- Cost Function:

$$J = \frac{1}{m}\sum_{i=1}^{m} L(y_i, \hat{y}_i) + \emptyset(w)$$

- Loss Function: $L(y_i, \hat{y}_i)$

$$-(y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i))$$

classification

$$sqrt(\frac{1}{k}\sum_{i=1}^{k}(y_i - \hat{y}_i)^2)$$

regression

RESULTS

Benchmark Information - 16nm GPU chips: Volta-IP1 and Xavier-IP2

- Local features are wrapped with zero-padding (only for FC)
- Approximately 90% of the samples for training and validation
- Approximately 10% of the samples for inference.

Models were developed in Python using TensorFlow and NumPy libraries.

Models were run on a cloud-based system with 2 CPUs, 2 GPUs and 32GB memory.

GPU	No. of Features	No. of Train Samples	No. Inference Samples
Volta-IP1	323	16500	1500
Xavier-IP2	239	2500	500

RESULTS

Dataset	Model-Architecture	Train Accuracy (%)	Inference Accuracy (%)	Train Time (minutes)	MAE (mV)
Volta-IP1 + Xavier-IP2	Classification-Flattened FC	94.5	94.5	10	7.30
	Classification-Hybrid FC	96.0	96.0	3	6.90
	Classification-NLP	92.6	92.6	80	7.46
	Regression-Flattened FC	98.0	93.0	9	7.79
	Regression-Hybrid FC	98.0	96.0	3	7.25
	Regression-NLP	95.0	95.0	90	7.28

Average run-time or prediction time

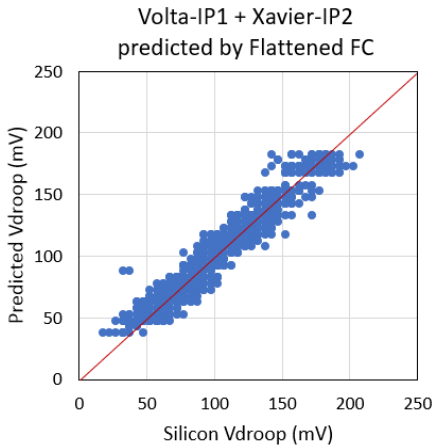
- For a 500-pattern set

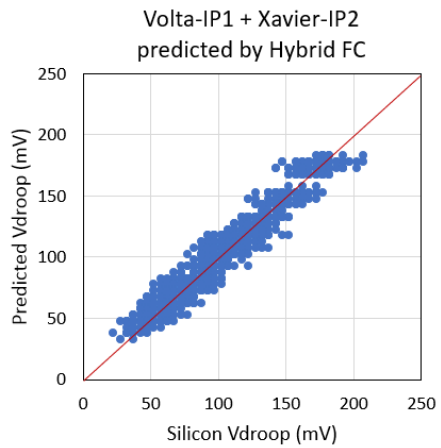
Method	**Run-Time**
Pre-Silicon Simulation	416 days
Post-Silicon Validation	84 mins
Proposed	0.33 secs

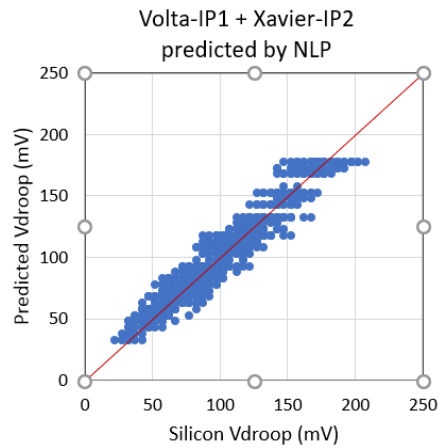
RESULTS

Correlation between the predicted and the silicon-measured V_{droop}

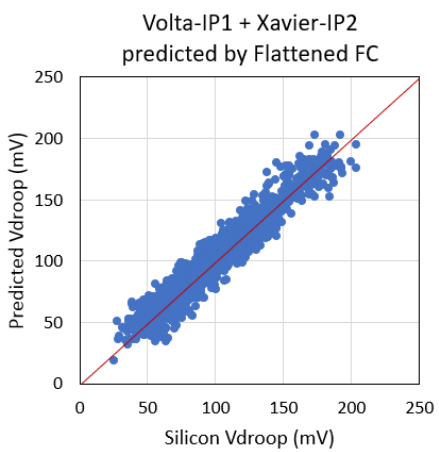
Classification

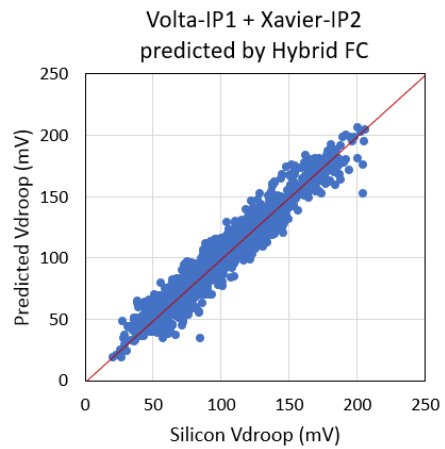


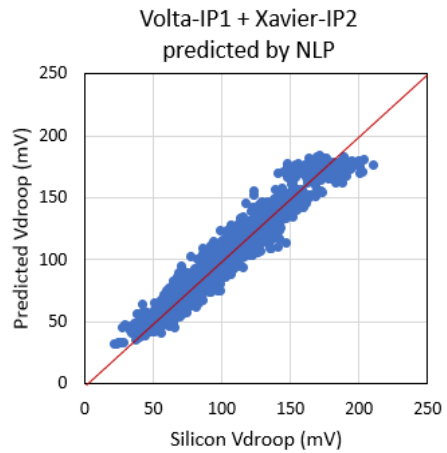




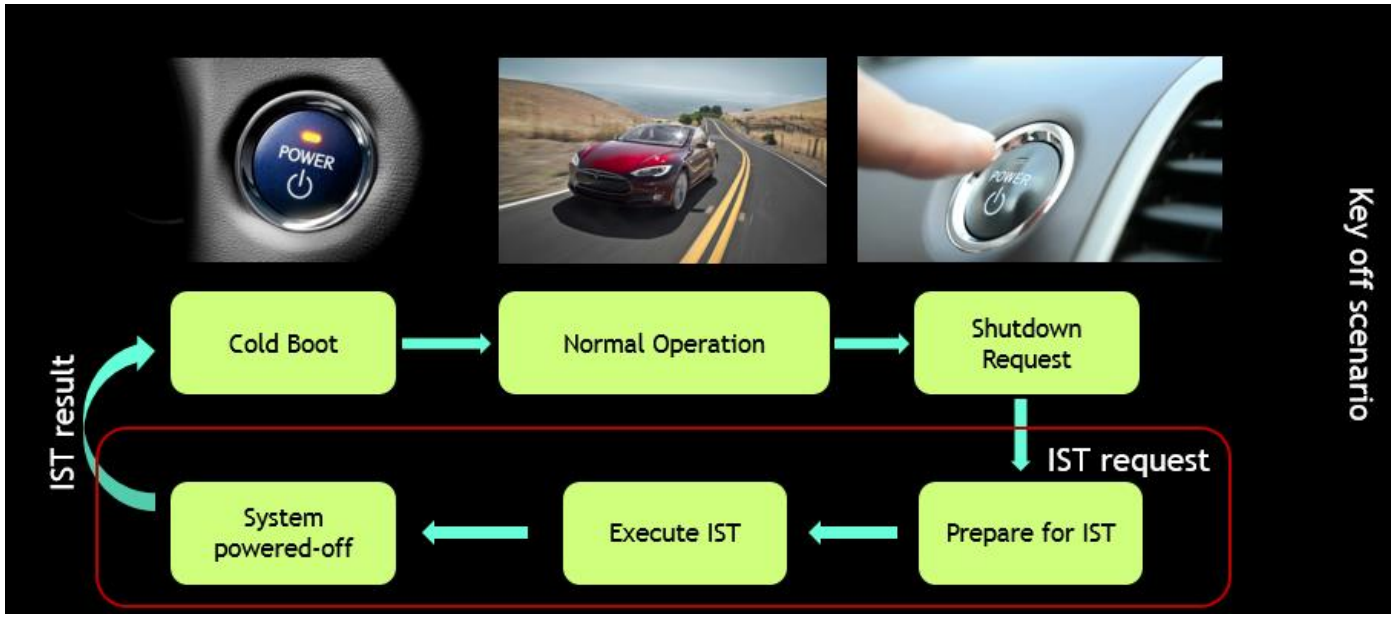
Regression







FUTURE WORK



- Train and apply DL for in-field test vectors noise estimation
- Shift Noise prediction
- Additional physical parameters
- Other architectures

RTL-Level Power Estimation Using Machine Learning

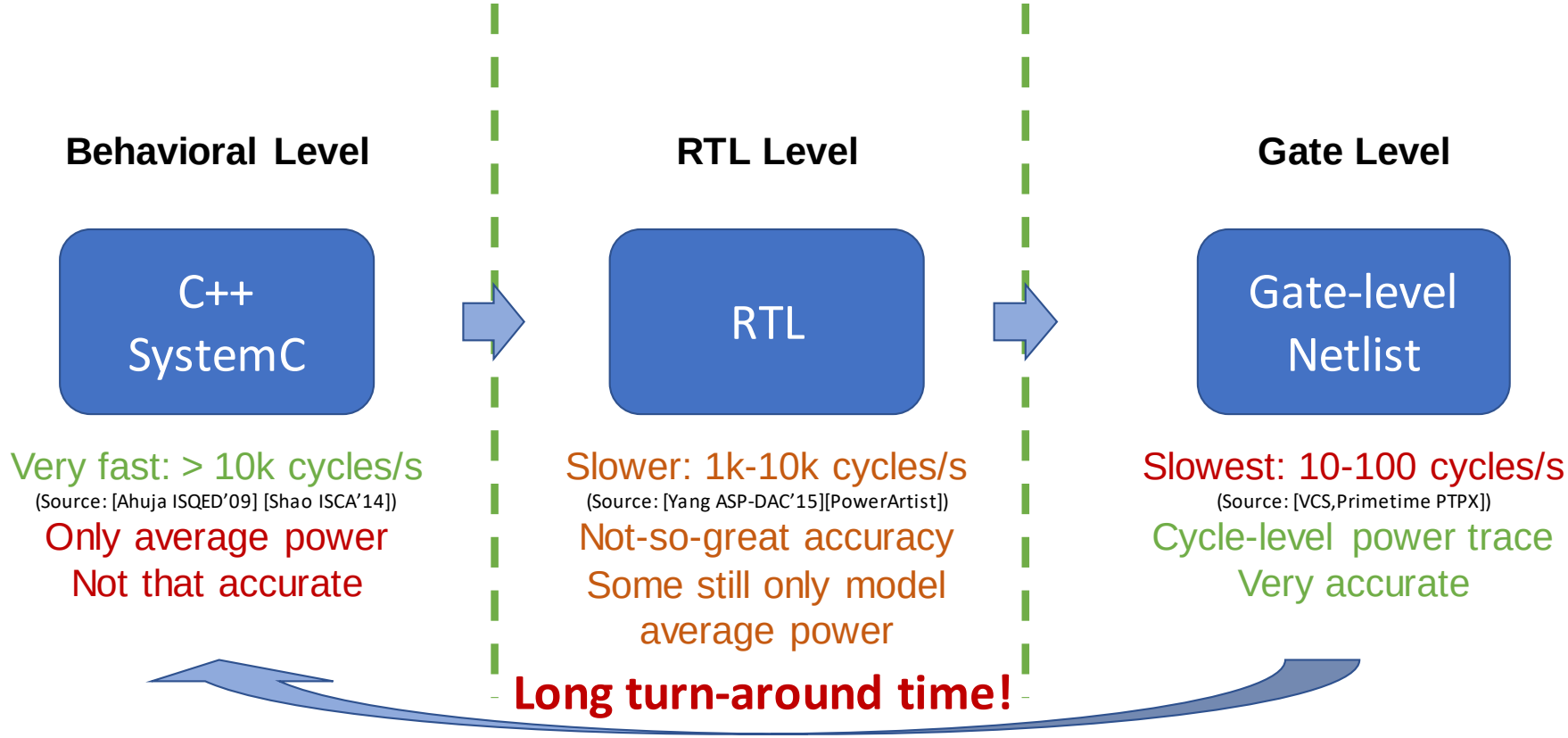
Mark Ren, Yan Zhang, Ben Keller, Brucek Khailany

Yuan Zhou, Zhiru Zhang

Cornell University

MOTIVATION

- Power modeling is either slow or inaccurate.
- Get power with accurate power estimation using simulation traces at early design stages?



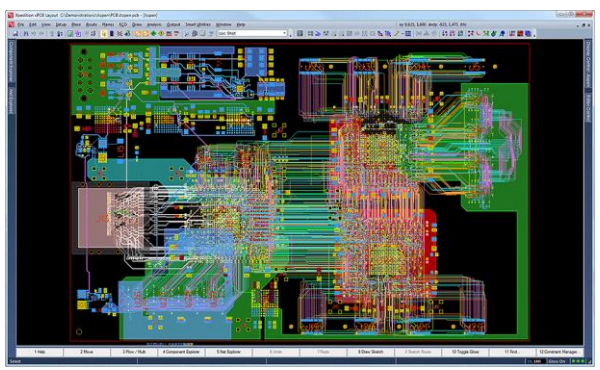
[Ahuja ISQED'09] S. Ahuja, D. A. Mathaikutty, G. Singh, J. Stetzer, S. K. Shukla, and A. Dingankar. "Power estimation methodology for a high-level synthesis framework." In Quality of Electronic Design, 2009. ISQED 2009. Quality Electronic Design, pp. 541-546. IEEE, 2009.
[Shao ISCA'14] Y. Shao, B. Reagen, G.-Y. Wei, and D. Brooks. "Aladdin: A pre-RTL, power-performance accelerator simulator enabling large design space exploration of customized architectures." In 2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA).
[Yang ASP-DAC'15] J. Yang, L. Ma, K. Zhao, Y. Cai, and T.-F. Ngai. "Early stage real-time SoC power estimation using RTL instrumentation." In Design Automation Conference (ASP-DAC), 2015 20th Asia and South Pacific, pp. 779-784. IEEE, 2015.
[PowerArtist] https://www.ansys.com/products/semiconductors/ansys-powerartist
[VCS] https://www.synopsys.com/verification/simulation/vcs.html
[Primetime PTPX] https://news.synopsys.com/index.php?item=123041

NVIDIA

OPPORTUNITY: ML FOR EDA

- Emerging field using Machine Learning for Electronic Design Automation (EDA) tasks
- Utilize GPU proficiency in ML tasks + find a way to map EDA applications to fit ML
- → Use machine learning / deep learning techniques to accurately estimate power at higher design abstraction level (RTL)
  - Shorter turn-around time, faster power validation, covers a diverse range of different workloads

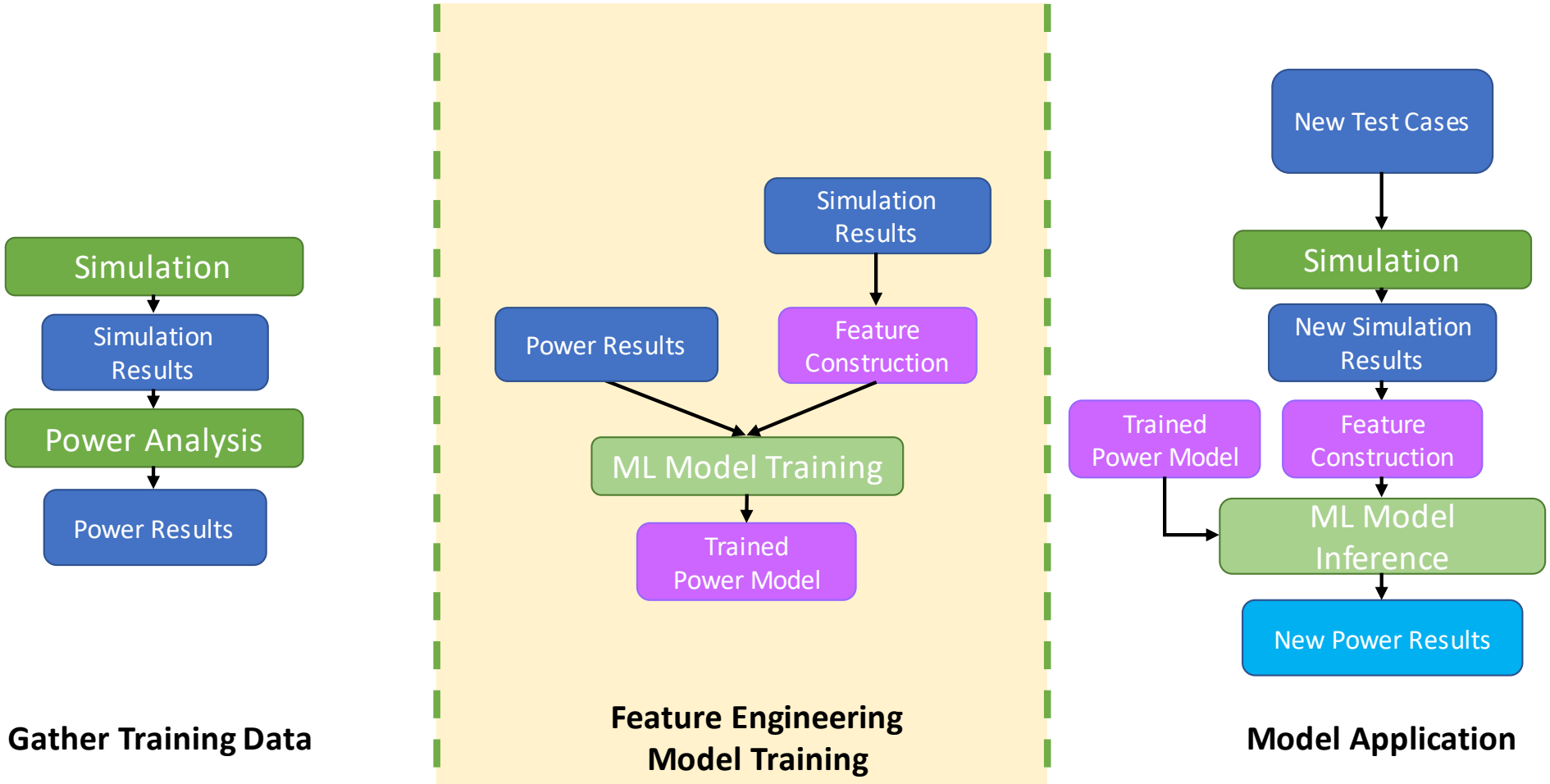
Source: https://towardsdatascience.com/



Source: https://roboticsandautomationnews.com/

NVIDIA

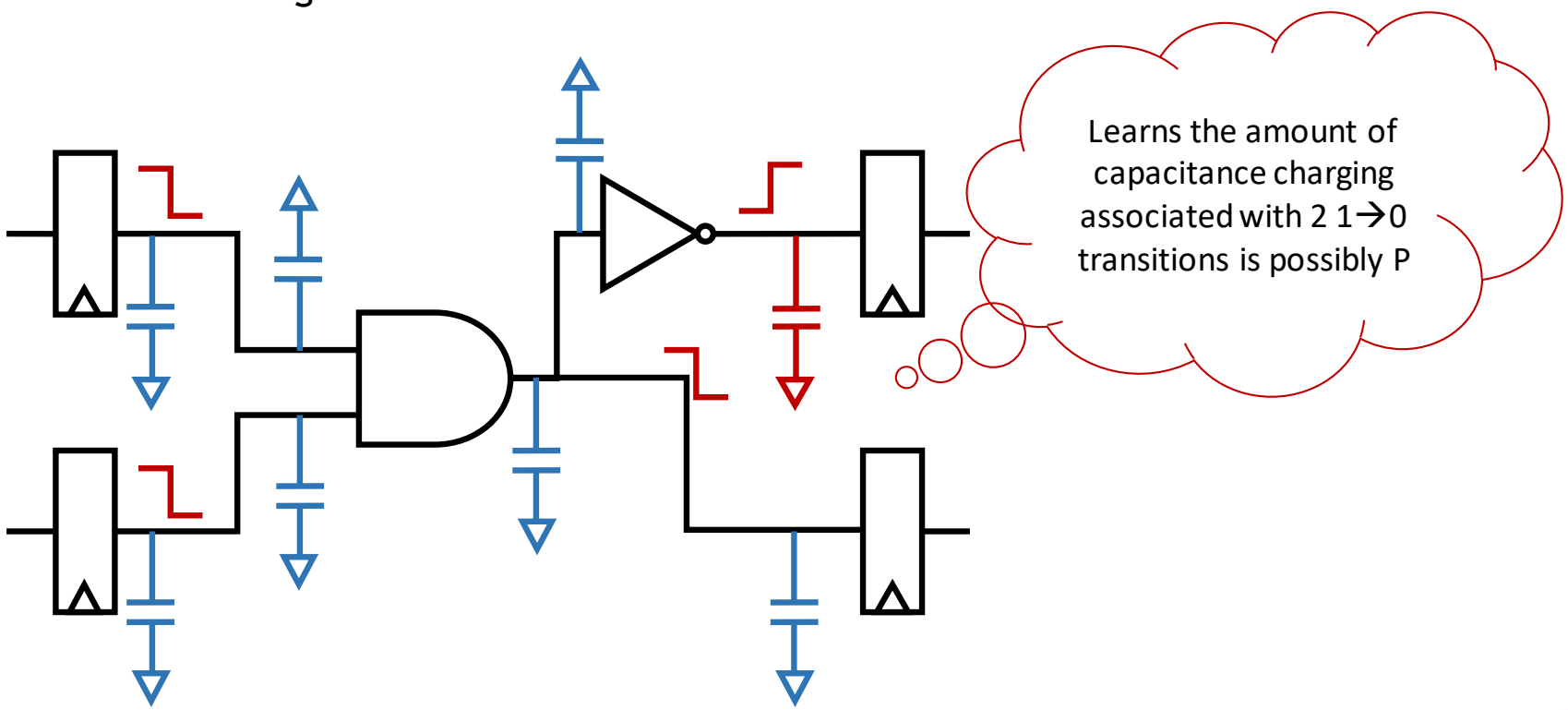
PROPOSED SOLUTION: ML-BASED POWER ESTIMATION WORKFLOW



NVIDIA

POWER ESTIMATION: CIRCUIT PERSPECTIVE

- Our models are essentially learning the switching capacitance associated with certain register switching activities
- Figuring out which caps switch and by how much is inhumanely complex and non linear
- → Perfect for machine learning!
- Example:



$$P = CV^2f$$

NVIDIA

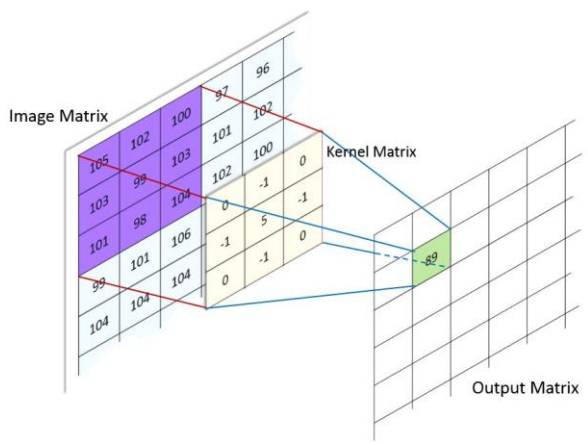
MODEL SELECTION

- Traditional ML: linear model, XGBoost
  - With principal component analysis (PCA) applied for overfitting avoidance
  - Pros: smaller model, faster training
  - Cons: Hard to capture non-linearities
- DL: convolutional neural net (CNN), multi-layer perceptron (MLP)
  - Pros: good for all sorts of non-linear models, good scalability
  - Cons: large model, longer training times, scalable but at a large startup cost (lots of parameters/nodes)

$$P = a_0 + a_1x_1 + a_2x_2 + a_3x_3 + \cdots anxn$$

$$\begin{pmatrix} P_1 \\ P_2 \\ \cdots \\ P_m \end{pmatrix} = \begin{pmatrix} a_1 \\ a_2 \\ \cdots \\ a_n \end{pmatrix} \cdot \begin{pmatrix} \vec{x}_1 \\ \vec{x}_2 \\ \cdots \\ \vec{x}_m \end{pmatrix}$$

Linear regression model



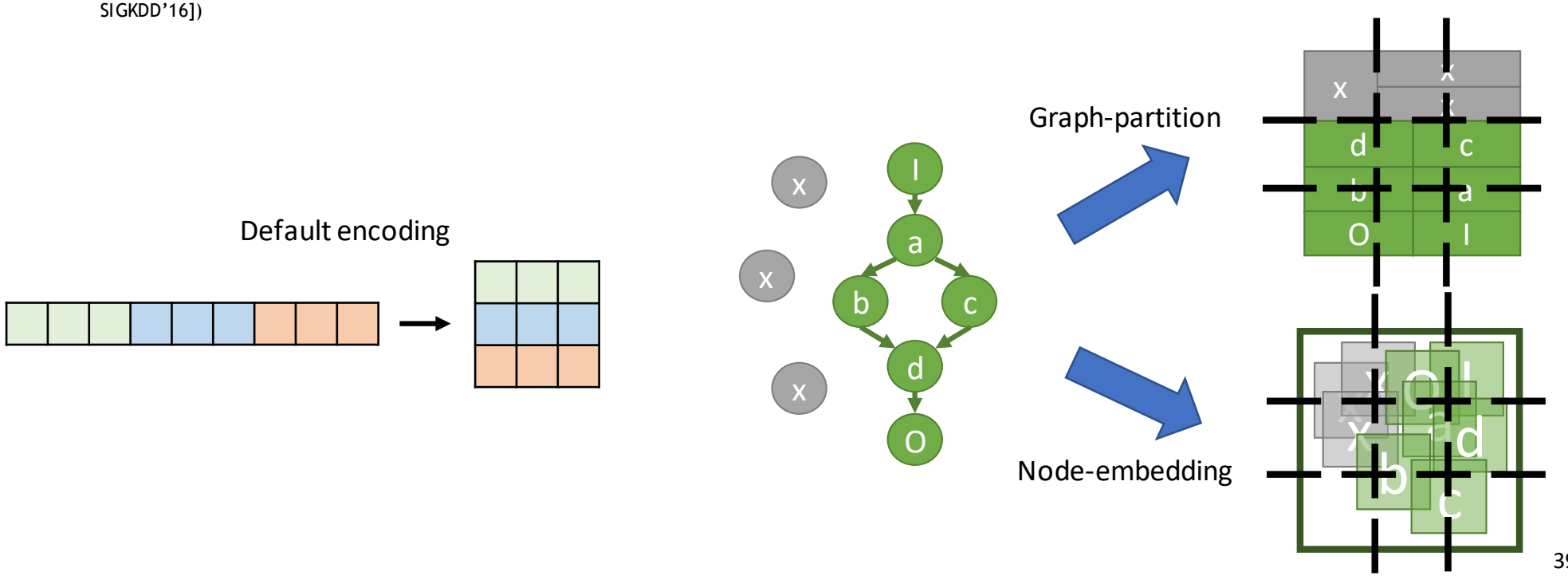
Source:https://brilliant.org/wiki/convolutional-neural-network/

CNN

NVIDIA

FEATURE CONSTRUCTION

- What information to use?
  - Register 0/1 state as inputs into model
- How to encode? CNNs work best when features have spatial relationship for their inputs
  - Default (naïve) encoding: random placement of register traces in CNN input
  - Graph-partition based: treat register relations as a graph, then partition to determine input placement
  - Node-embedding based: Use `node2vec` to convert graph nodes into embeddings (Source: [Grover SIGKDD'16])



[Grover SIGKDD'16] Grover, Aditya, and Jure Leskovec. "node2vec: Scalable feature learning for networks." In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855-864. ACM, 2016.

EXPERIMENT SETUP

Test Designs

Design	Description	Register + I/O signal count	Gate count	PTPX throughput (cycles/s)	Training set (# cycles)	Test set (# cycles)
qadd_pipe	32-bit fixed point adder	160	838	1250	Random stimulus (480k)	Random stimulus (120k)
qmult_pipe{1, 2, 3}	32-bit fixed point multiplier with 1, 2, or 3 pipeline stages	{384, 405, 438}	{1721, 1718, 1749}	{144.9, 135.1, 156.3}	Random stimulus (480k)	Random stimulus (120k)
float_adder	32-bit floating point adder	381	1239	714.3	Random stimulus (480k)	Random stimulus (120k)
float_mult	32-bit floating point multiplier	372	2274	454.5	Random stimulus (480k)	Random stimulus (120k)
NoCRouter	Network-on-chip router for a CNN accelerator	5651	15076	44.7	Unit-level testbenches (910k)	Convolution tests (244k)
RISC-V Core	RISC-V Rocket Core (`SmallCore`)	24531	80206	45	RISC-V ISA tests (2.2M)	RISC-V benchmarks (1.7M)

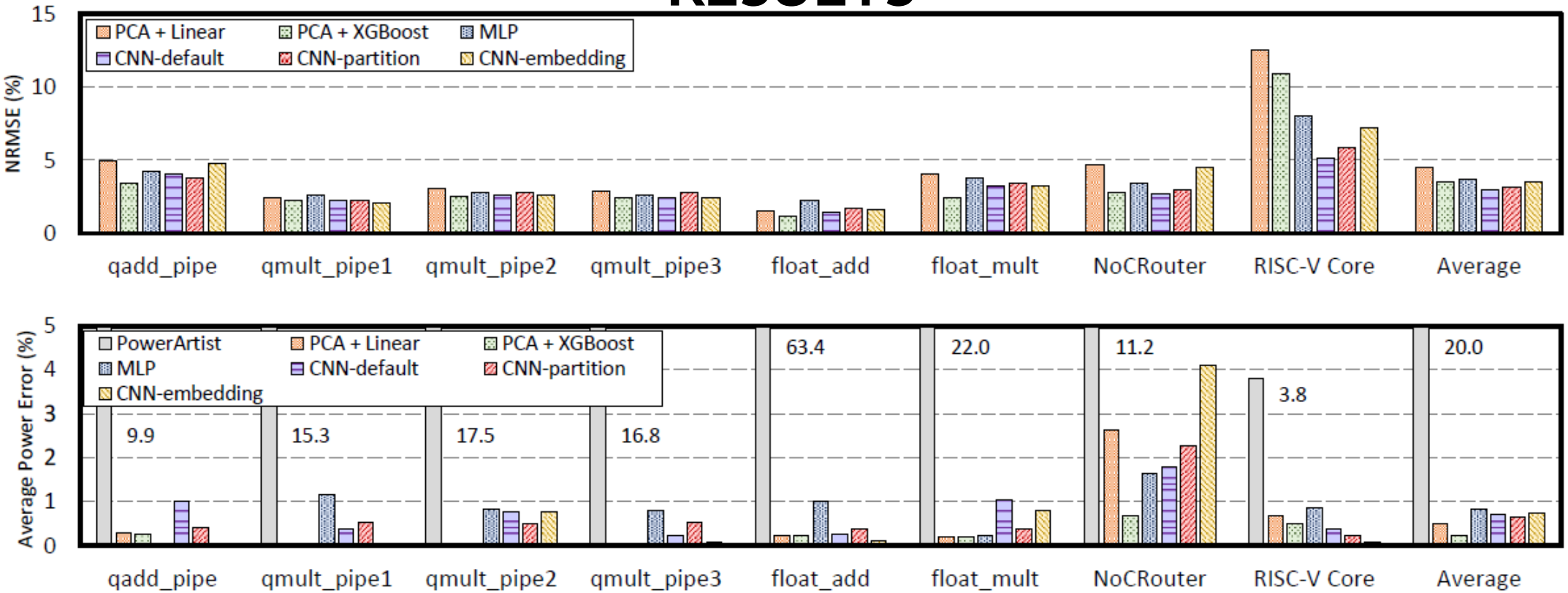
Source: Y. Zhou, et. al "PRIMAL: Power Inference using Machine Learning", to appear in DAC 2019, June

- Normalized Root Mean Square Error (NRMSE)
  - $NRMSE = RMSE/\bar{y}$
  - Cycle-by-cycle basis
- Directly look at the power traces to see how good it fits
  - Good for catching outliers
  - Cycle-by-cycle basis

EXPERIMENT SETUP

- ML training and inference infrastructure:
  - NVIDIA 1080Ti GPU
  - Software packages: *network*, *metis*, *node2vec*, *Python 3.5*, *Keras 2.1.6*, *scikit-learn*, *xgboost 0.72.1*
- Ground truth and comparison baseline gate level power analysis infrastructure
  - Intel Xeon CPU server, 64GB RAM

RESULTS

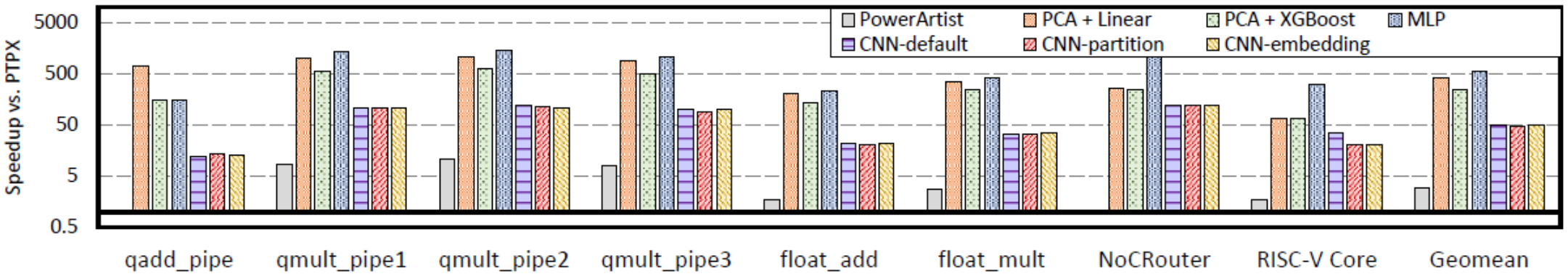


Source: Y. Zhou, et. al "PRIMAL: Power Inference using Machine Learning", to appear in DAC 2019, June

- Good accuracy
  - <5% average power estimation for all test cases
  - CNNs outperform linear models for bigger designs
  - Accuracy outperforms commercial tool

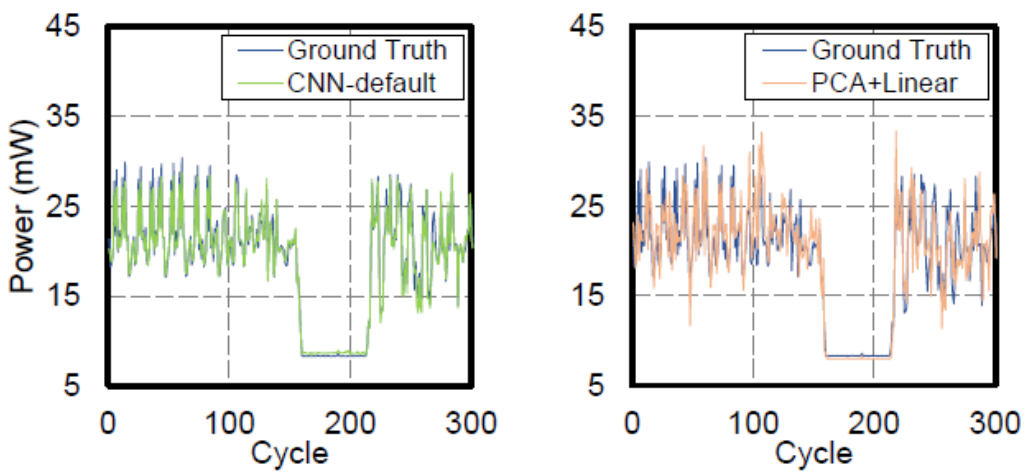
NVIDIA

RESULTS



Source: Y. Zhou, et. al "PRIMAL: Power Inference using Machine Learning", to appear in DAC 2019, June

- ~50X speedup against gate simulation + power analysis



300 cycles of RISCV core dhrystone benchmark

Source: Y. Zhou, et. al "PRIMAL: Power Inference using Machine Learning", to appear in DAC 2019, June

- Cycle-by-cycle traces show better accuracy for CNNs compared to linear models

NVIDIA

CONCLUSIONS

- We can get both good accuracy and high speedup with ML-based power estimation
- Achieves ~50X speedup over baseline with <5% error
- A good example of using ML for EDA purposes
- GPUs greatly benefit training/inference time in ML for EDA


Thank You!
NVIDIA