Machine Learning for Design Efficiency

Elyse Rosenbaum

University of Illinois at Urbana-Champaign

IEEE CAS CAS Dallas Chapter IOWA STATE UNIVERSITY OF SCIENCE AND TECHNOLOGY UT DALLAS TEXAS A&M UNIVERSITY WASHINGTON STATE UNIVERSITY TEXAS INSTRUMENTS

“Machine Learning” is > 60 years old

A. L. Samuel

Some Studies in Machine Learning Using the Game of Checkers

Abstract: Two machine-learning procedures have been investigated in some detail using the game of checkers. Enough work has been done to verify the fact that a computer can be programmed so that it will learn to play a better game of checkers than can be played by the person who wrote the program. Furthermore, it can learn to do this in a remarkably short period of time (8 or 10 hours of machine-playing time) when given only the rules of the game, a sense of direction, and a redundant and incomplete list of parameters which are thought to have something to do with the game, but whose correct signs and relative weights are unknown and unspecified. The principles of machine learning verified by these experiments are, of course, applicable to many other situations.

IBM JOURNAL • JULY 1959

The View from Popular Culture

- ML not distinguished from AI
- Intelligent machines
  - With beyond-human capabilities
- Overwhelmingly identified with neural nets
  - As far back as 1968

This Talk

- Is not about sentient machines
- Does not assume that machine learned models are necessarily neural networks

Limits of Modern EDA

- Design respins have not been eliminated
- Many of the observed failures during qualification testing are the direct result of an insufficient modeling capability
  - Sources of such failures include mistuned analog circuits, signal timing errors, reliability problems, and crosstalk [1]
  - Variability cannot be modeled in a manner that is both accurate and computationally efficient
- Simulation-based design optimization has had only limited success
  - Simulation “in-the-design-loop” often too slow and leads to impractical designs
- Proposal: use machine learning algorithms to overcome those hurdles!

[1] Harry Foster, “2012 Wilson Research Group Functional Verification Study,” http://www.mentor.com/products/fv/multimedia/the-2012-wilson-research-group-functional-verification-studyview

What does ML have to offer to us?

- We (i.e., circuit designers and semiconductor device specialists) often need to minimize a non-convex function
  - Model fitting
  - Design optimization
- We need to construct stochastic models
  - Process variations
- ML specialists perform similar numerical analyses, and we can repurpose their code

Example: Model Fitting

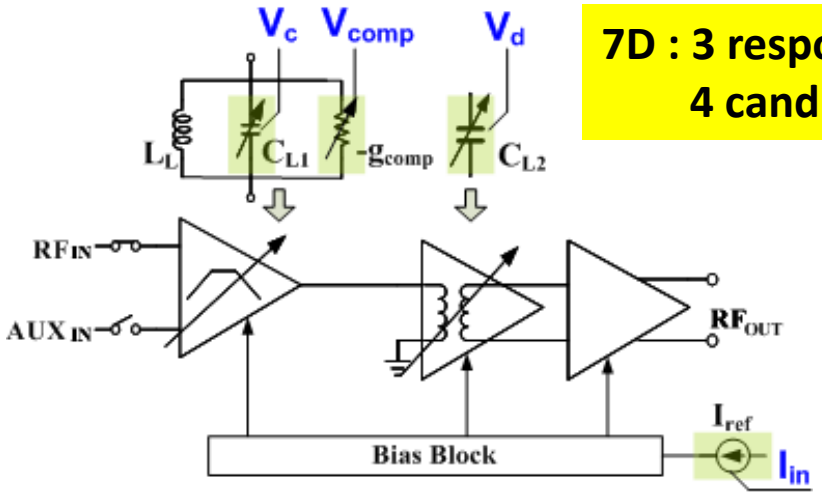
- Fit $I = \mathbf{I_s} \cdot \exp\left\{\frac{\mathrm{q}V}{\mathbf{n}\mathrm{kT}}\right\} = f(V, \mathbf{I_s}, \mathbf{n})$
- Estimate parameters by fitting model to the measurement data

$$\underset{\mathbf{I_S},\,\mathbf{n}}{\mathbf{argmin}} \sum_{j=1}^{m} \left(f\left(V_j, \mathbf{I_s}, \mathbf{n}\right) - I_j\right)^2$$

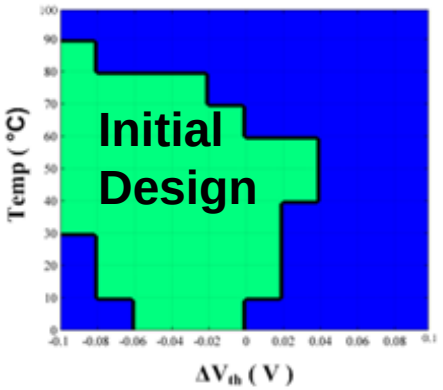
  - This optimizes the model
- The loss function is **nonconvex**
  - No analytic solution
  - Must solve using iterative methods
  - Need a robust algorithm that avoids getting stuck at a local minimum and that converges quickly
  - ML practitioners have created open-source libraries with suitable algorithms, e.g. SGD, BFGS, conjugate gradient …

Examples: Circuit Design Optimization

Goal: Identify the optimal set of calibration knobs for this RF amp



7D : 3 responses,
4 candidate "knobs"

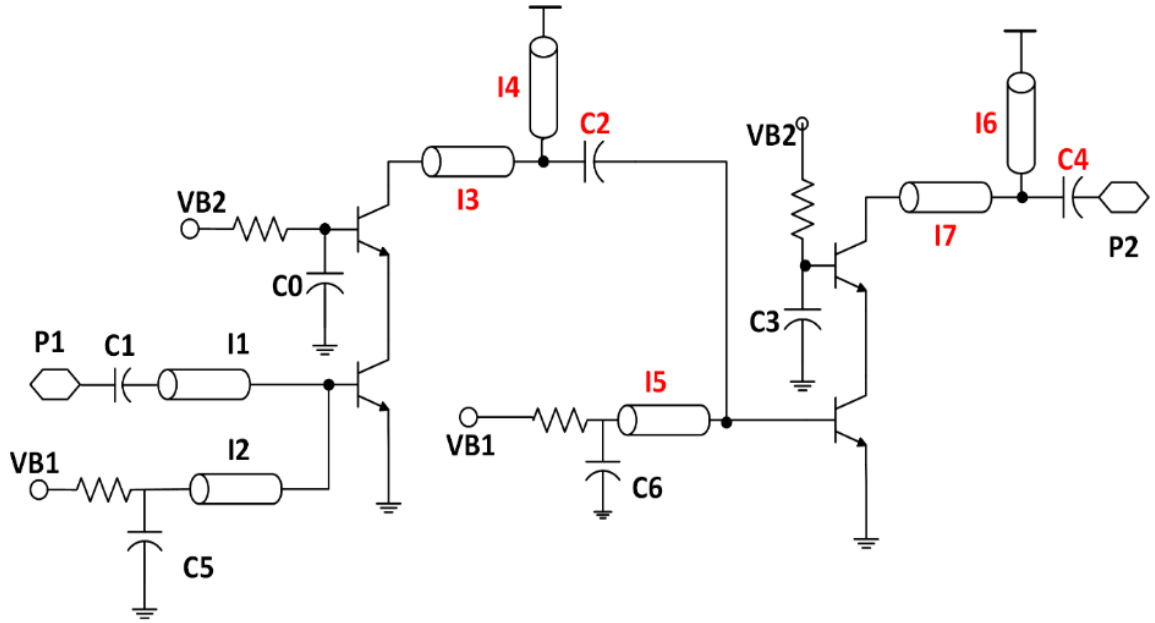


(a) Nominal design

Feasible sub-region

Infeasible sub-region

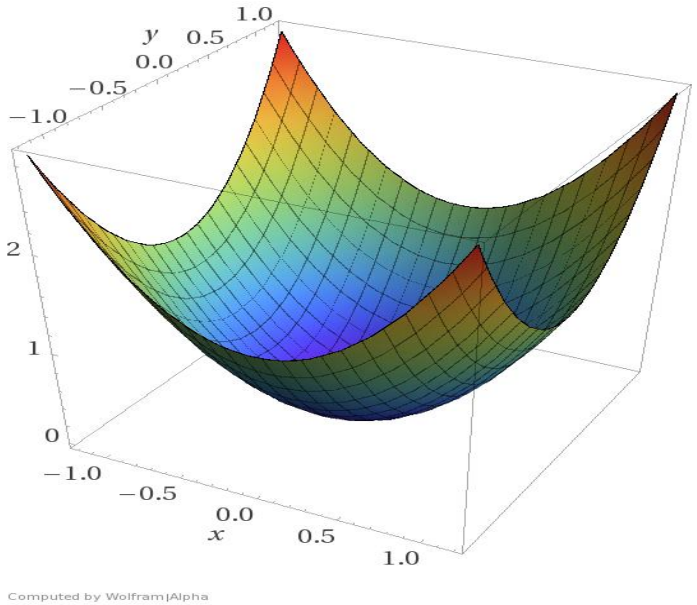
Goal: Optimize a power amplifier with 17 design parameters



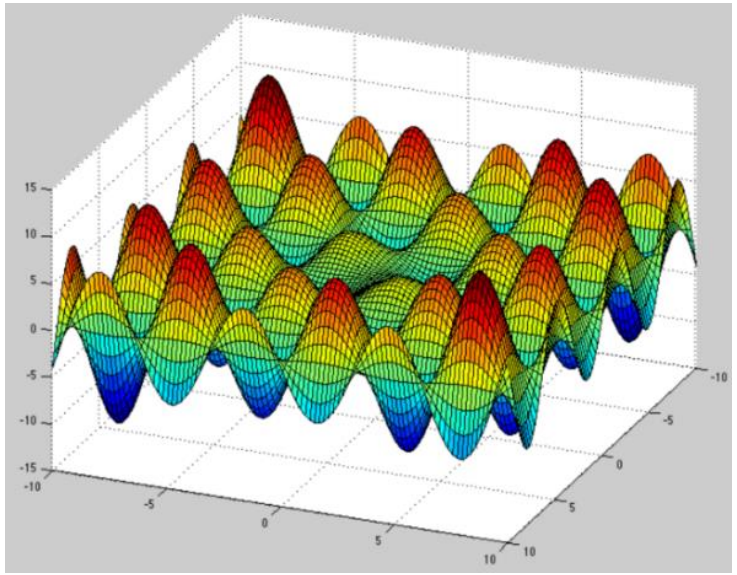
Design Optimization

- The loss function we want to minimize may be skew, power, area or some combination of those
  - Or we may wish to find the maximum of some objective function, e.g. gain
- Usual challenges: large number of design variables, non-convex loss function, gradient information unavailable

Response Surface (Dream)



Response Surface (Reality)

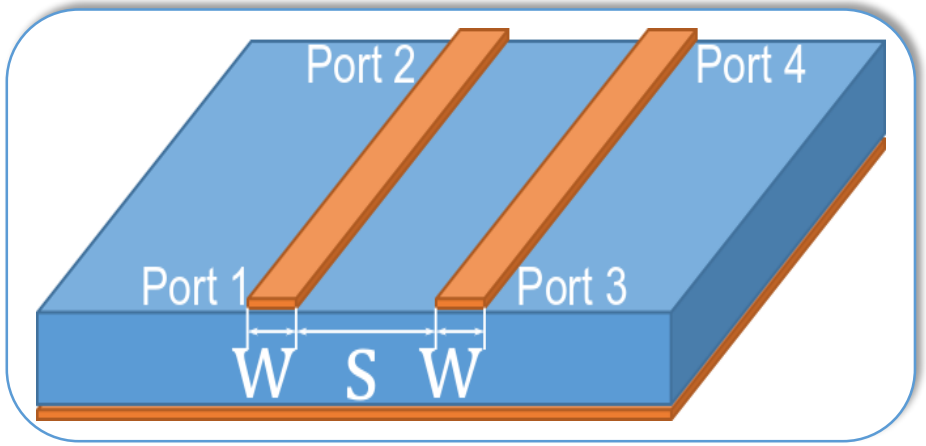


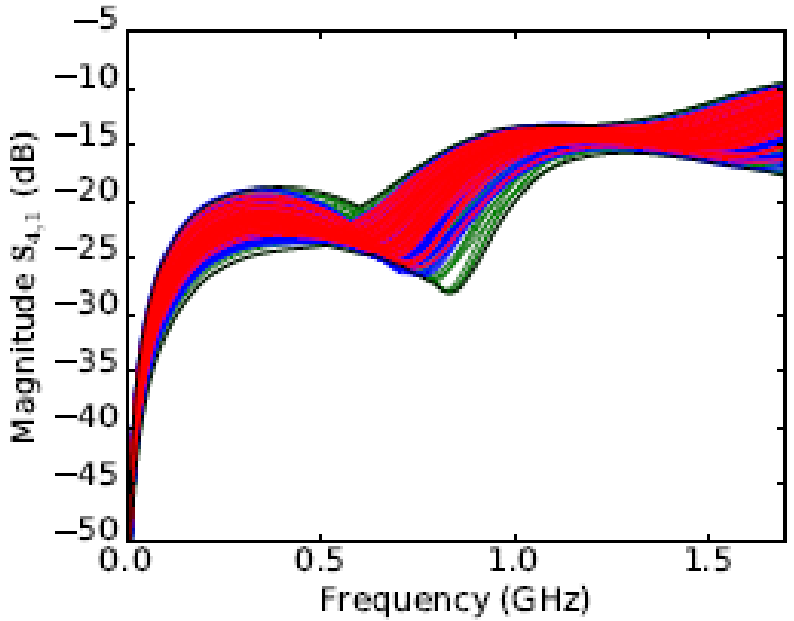
Generative Models

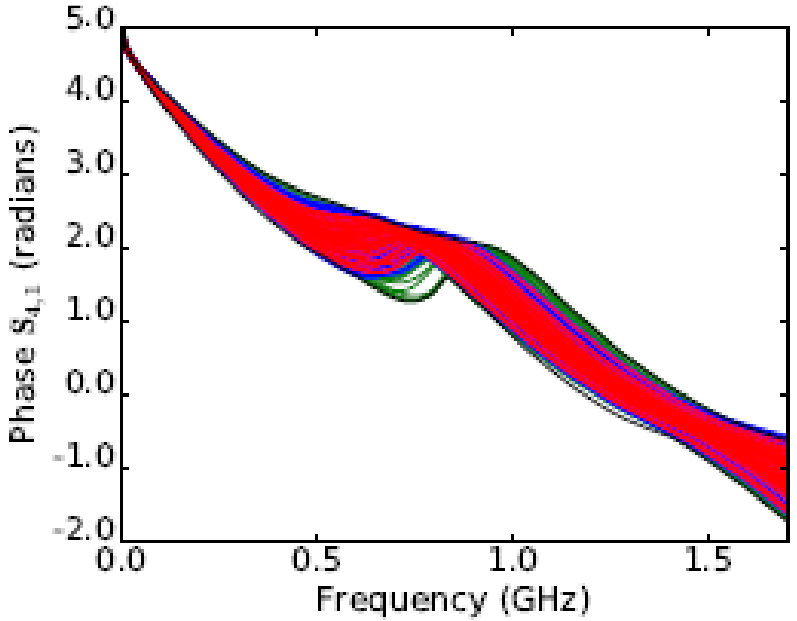
- A familiar model: $I_{DS} = \frac{k}{2}(V_{ov})^2$
  - This is a **discriminative model**. It provides E(*y*|*x*).
- The alternative is a **generative model**. It provides the joint probability distribution function f(*x*, *y*).
  - One may sample from this distribution.
- We need such models due to unavoidable manufacturing variations

Motivational Example

- S-parameters of coupled microstrip lines
- Line widths, spacing and dielectric permittivity vary
- Results shown below for S41 ($\sigma = 0.1 \cdot \bar{x}$)
  - 50 training samples (red); 500 generated samples (blue); 475 validation samples (green)
- Monte Carlo sampling for system yield analysis







S. De Ritter et al., in *EDAPS* 2016.

What does ML have to offer?

- One might estimate the joint pdf from the training data using a non-parametric model
  - E.g., kernel density estimation or k-nearest neighbor density estimation
  - The size of the model grows with the amount of training data, becoming computationally inefficient
- Parametric models are preferred
  - E.g., **neural networks**

State-of-the-art Generative Models

- Face generator: This Person Does Not Exist
  - Uses a generative adversarial network
  - The generator and discriminator are neural networks
    - A nice introduction to GAN may be found at https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29
- If the embedded link doesn't work, you can reach the demo at https://www.thispersondoesnotexist.com/

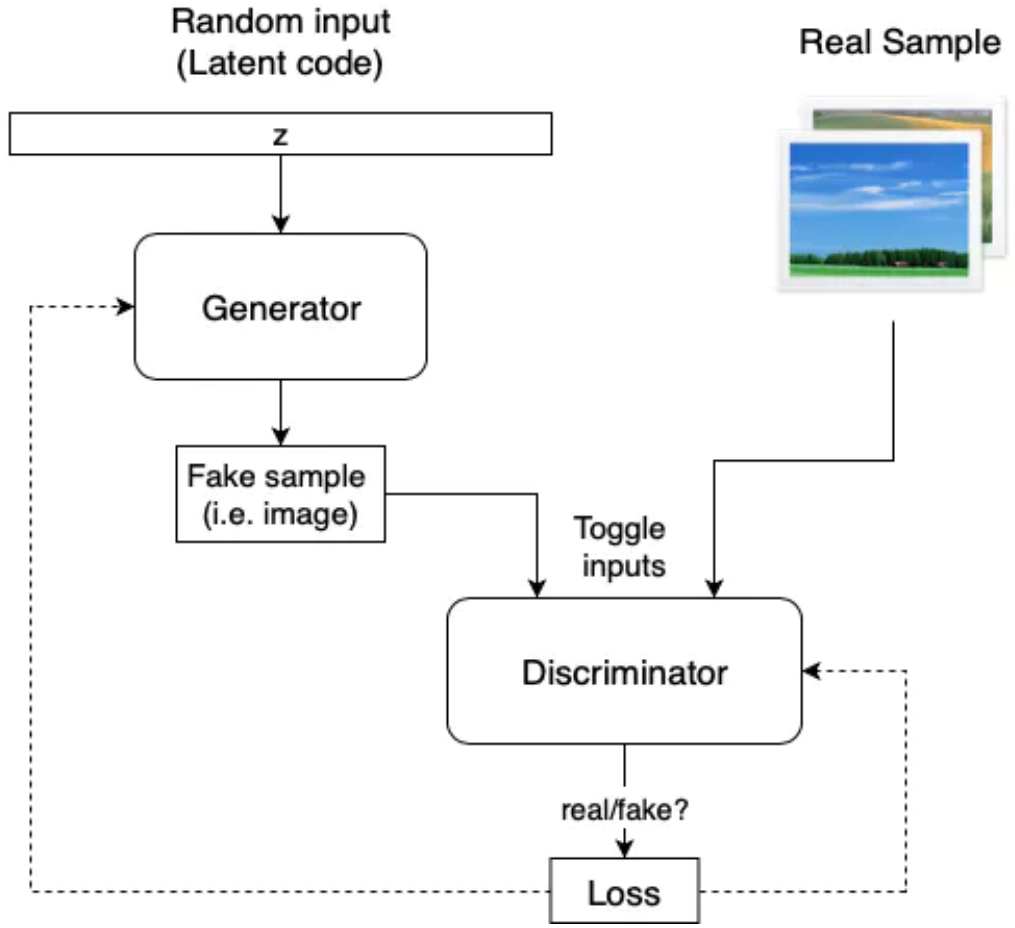
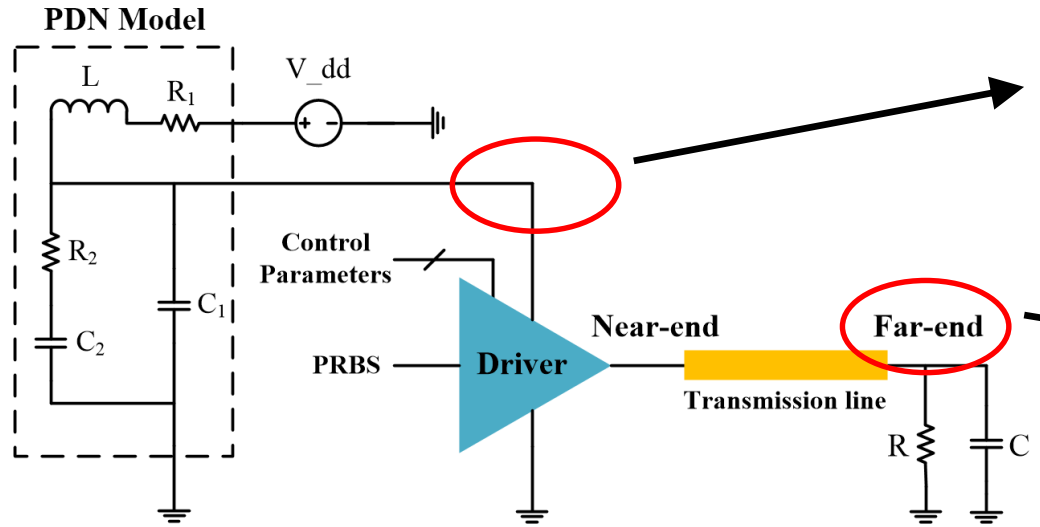


Figure: D. Paez, "This person does not exist is the best one-off website of 2019," inverse.com, 6/21/19.

Behavioral Models

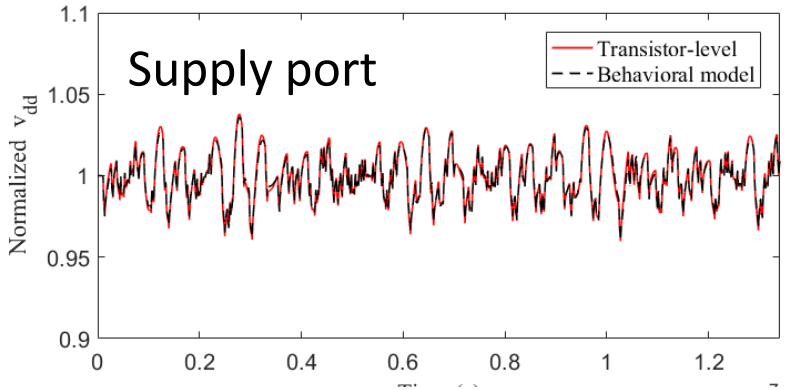
- Component models are used for pre-manufacturing verification of system design and for system yield analysis
  - Component may be an IP block or a complete IC
- A behavioral model will represent only the response of the component as seen at its external ports, i.e., where it connects to other components of the system
- A behavioral model may be less complex than a model that reveals the inner workings of the component
  - Computationally efficient
- By obscuring the inner workings of the component, the behavioral model protects the designer's IP
- Very important to validate behavioral models derived using ML
  - Does model correctly predict the response of the system to previously unseen stimuli?

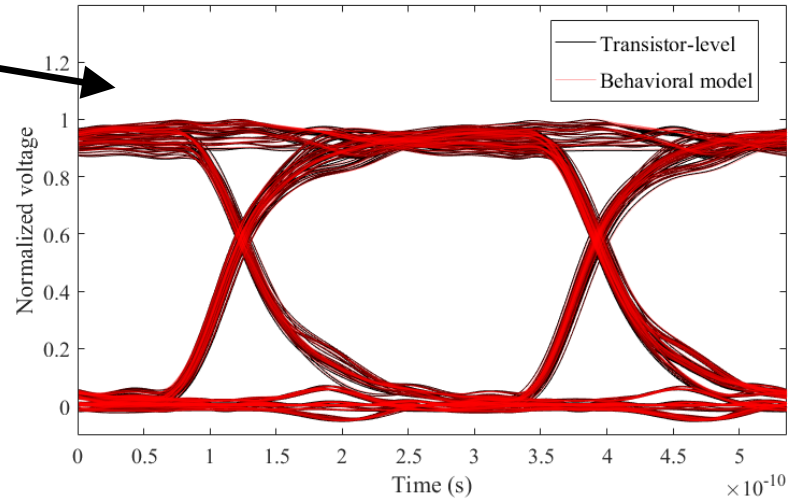
SI and PI Co-simulation using Behavioral Models



- Simulator
  - Hspice
- Load
  - TL = 50 mm
  - R = 60 Ohm
  - C = 0.7 pF

Transistor-level vs. Behavioral Model





M. Swaminathan, GT/CAEML

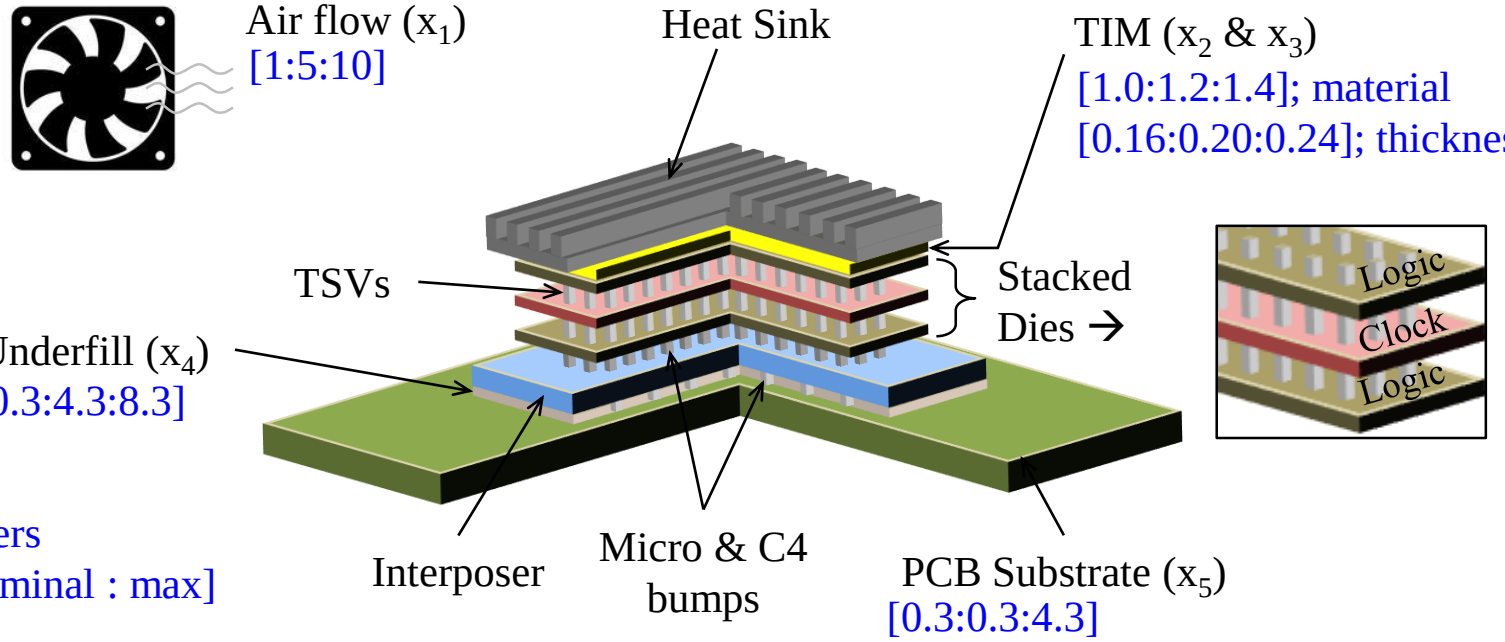
Applications of ML to EDA

- Design Optimization
  - Clock skew minimization for 3D-IC
  - IP Reuse
- Generative Modeling
  - Stochastic models of PCB interconnects
  - PPA prediction
- Circuit Macro-modeling
  - Recurrent neural network
- Data Analytics
  - Early detection of hardware failure
- Additional examples from CAEML What's that?

Center for Advanced Electronics through Machine Learning (CAEML)

- Vision statement
  - To enable fast, accurate design and verification of microelectronic circuits and systems by creating machine-learning algorithms to derive models used for electronic design automation
- An NSF Industry/University Cooperative Research Center
  - University of Illinois at Urbana-Champaign
    - Lead site. Prof. Elyse Rosenbaum, Center Director
  - Georgia Tech
    - Prof. Madhavan Swaminathan, Site Director
  - North Carolina State University
    - Prof. Paul Franzon, Site Director

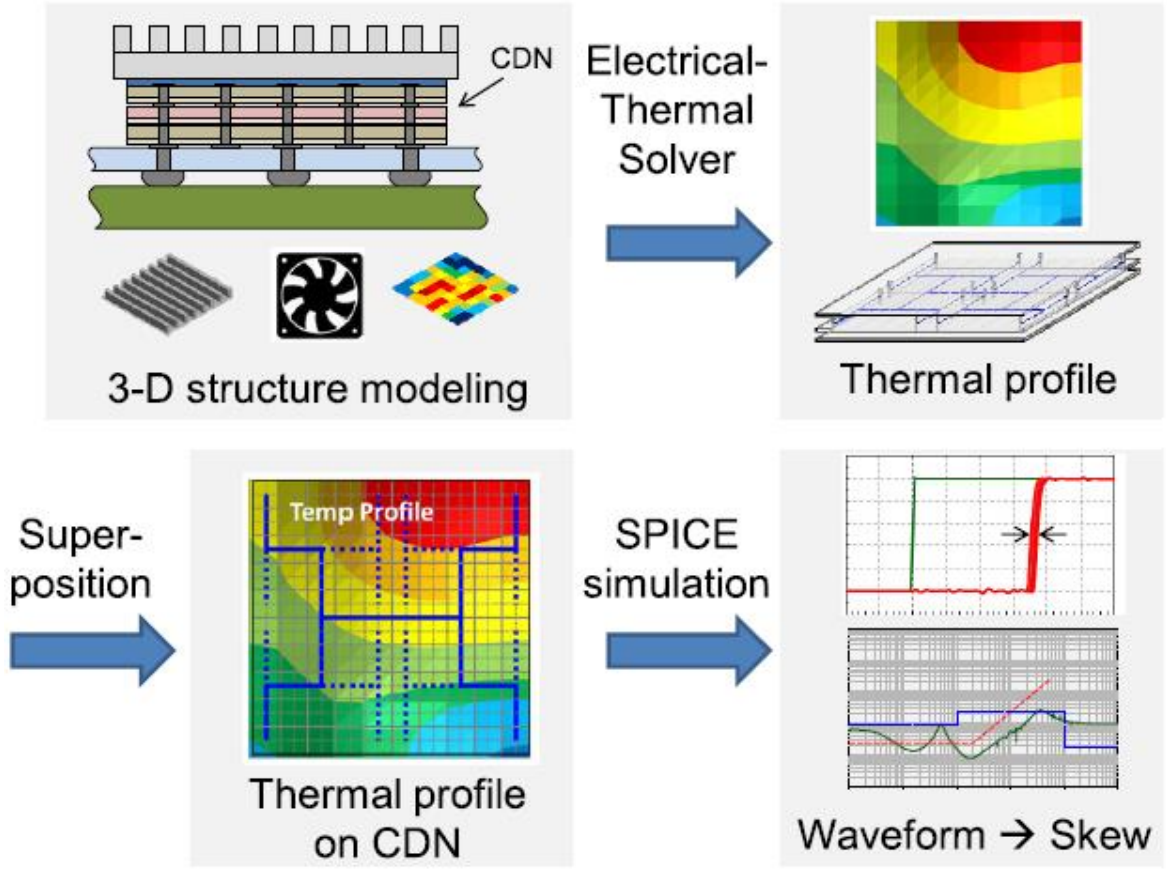
Thermal Design Optimization for 3D-IC



S. J. Park, B. Bae, J. Kim and M. Swaminathan, "Application of Machine-Learning for Optimization of 3-D Integrated Circuits and Systems," *IEEE Trans. VLSI,* June 2017.

- Clock Skew is affected by Temperature (magnitude and gradient)
- Temperature is controlled by FIVE input (or control) parameters
- Control parameters have certain constraints
- Objective is therefore to TUNE these parameters to minimize Skew
- More generally, seek to find $X_{opt} = \underbrace{argmin}_{\mathbf{x}}((f(X))$. Accurate modeling of f(X) needed only near min
- Use ML-based Bayesian Optimization

Full Design Space Exploration too Costly



- 3D (finite volume) simulations + SPICE-type circuit simulation
- Need to limit the number of designs that are simulated

Introduction to Bayesian Optimization

$$P(f|\boldsymbol{D}) = \frac{P(\boldsymbol{D}|f)P(f)}{P(\boldsymbol{D})} \propto P(\boldsymbol{D}|f)P(f)$$

Posterior — Likelihood — Prior Model

- Model *f* as a random process
  - Usually a GP
- Obtain next training sample $f(X_i)$, where $X_i = \underbrace{\boldsymbol{argmax}}_{\mathbf{x}} \{P(f(X_i) > f(X^+))\}$
  - Maximizes PI, probability of improvement
- Other acquisition functions
  - EI, expected improvement
  - UCB / LCB, and more …
  - Designed to balance exploration (high variance) and exploitation (high mean)

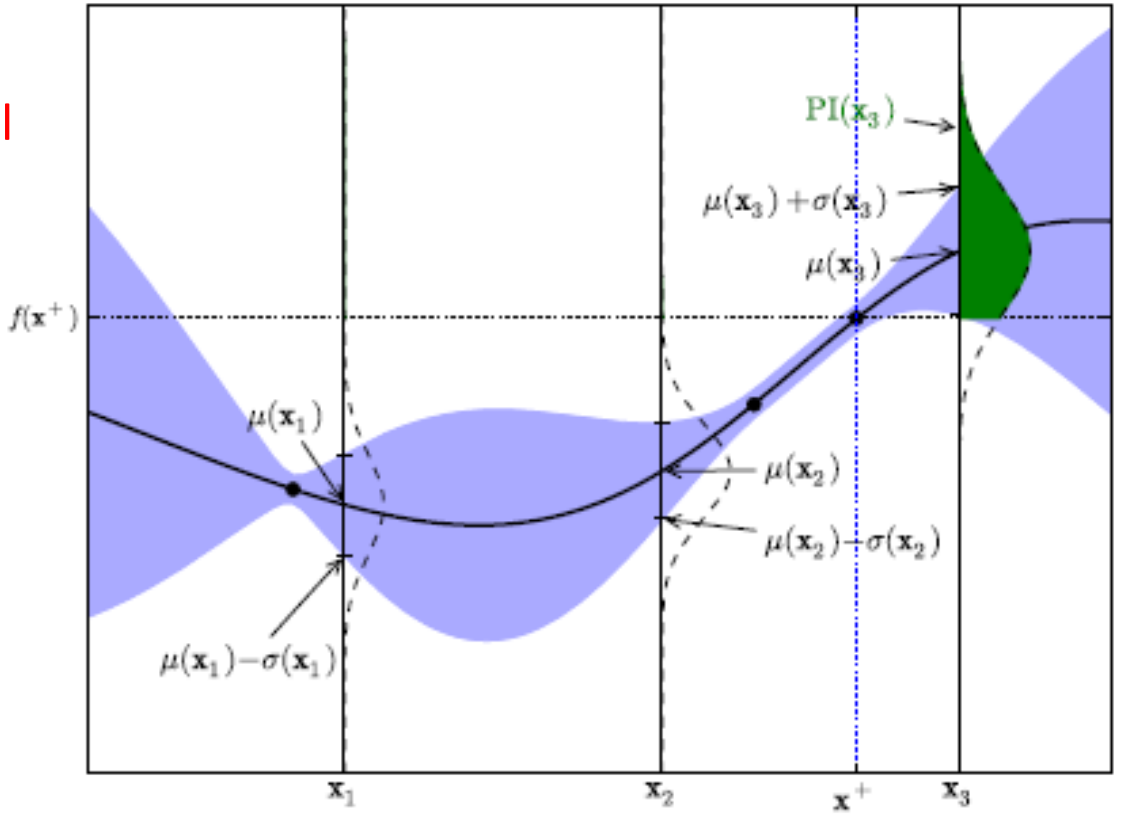
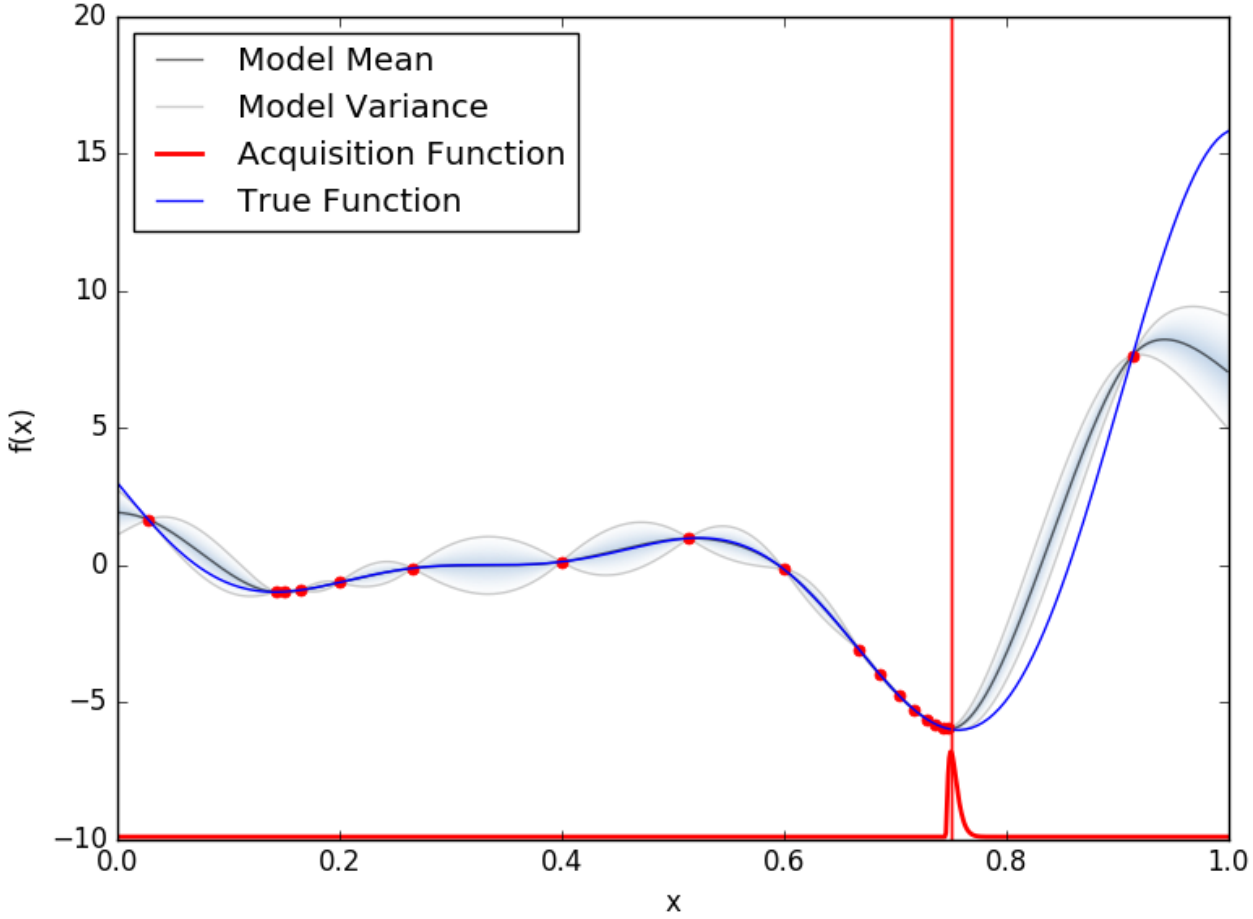


Figure: E. Brochu. arXiv:1012.2599

1-D Demo of Bayesian Optimization

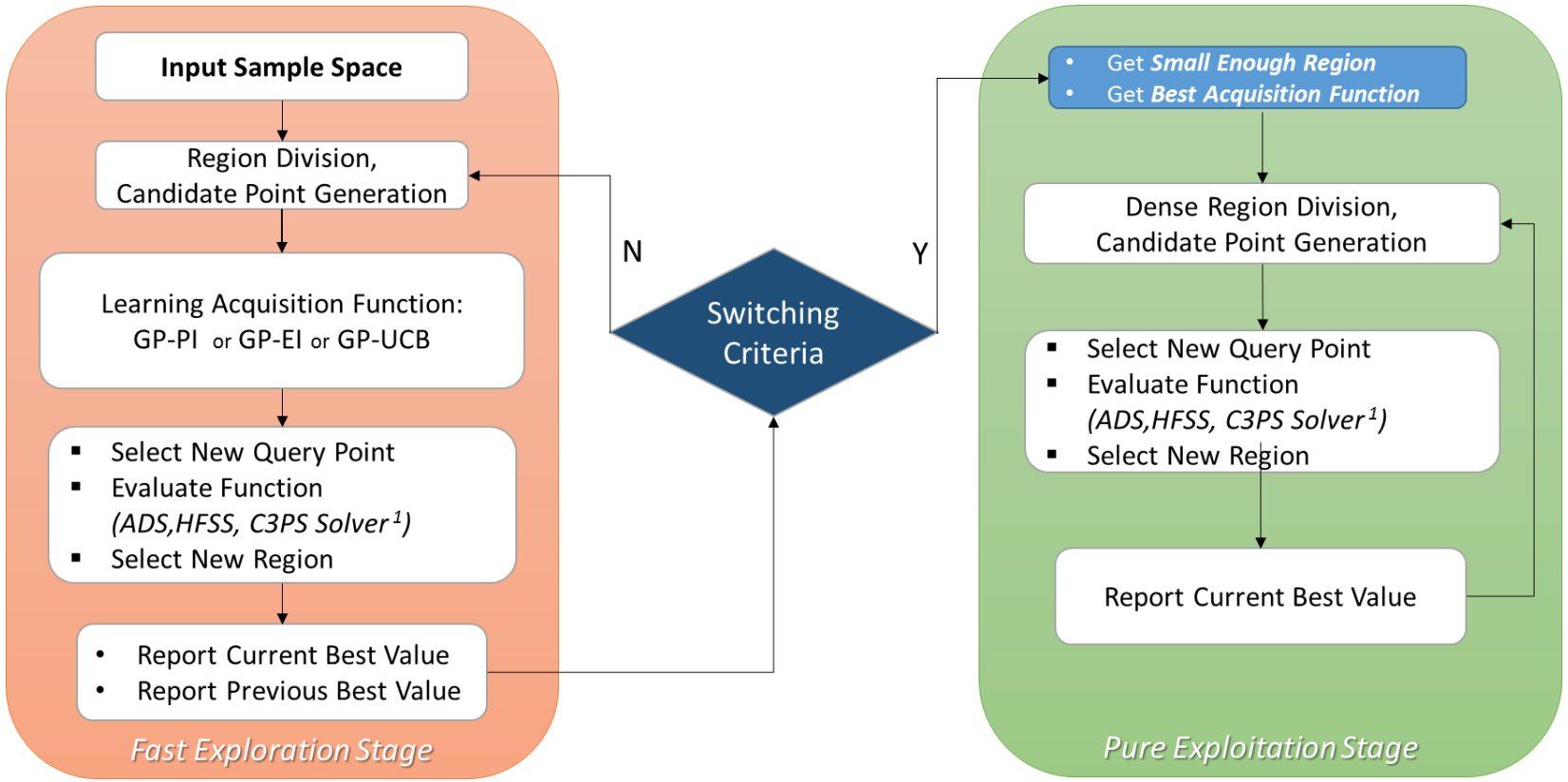


Plot generated by P. Franzon (CAEML/NCSU) using python GPyOpt package

Done!

True function: $f(x) = (6x-2)^2 \sin(12x-4) \quad where\ x \in [0,1]$

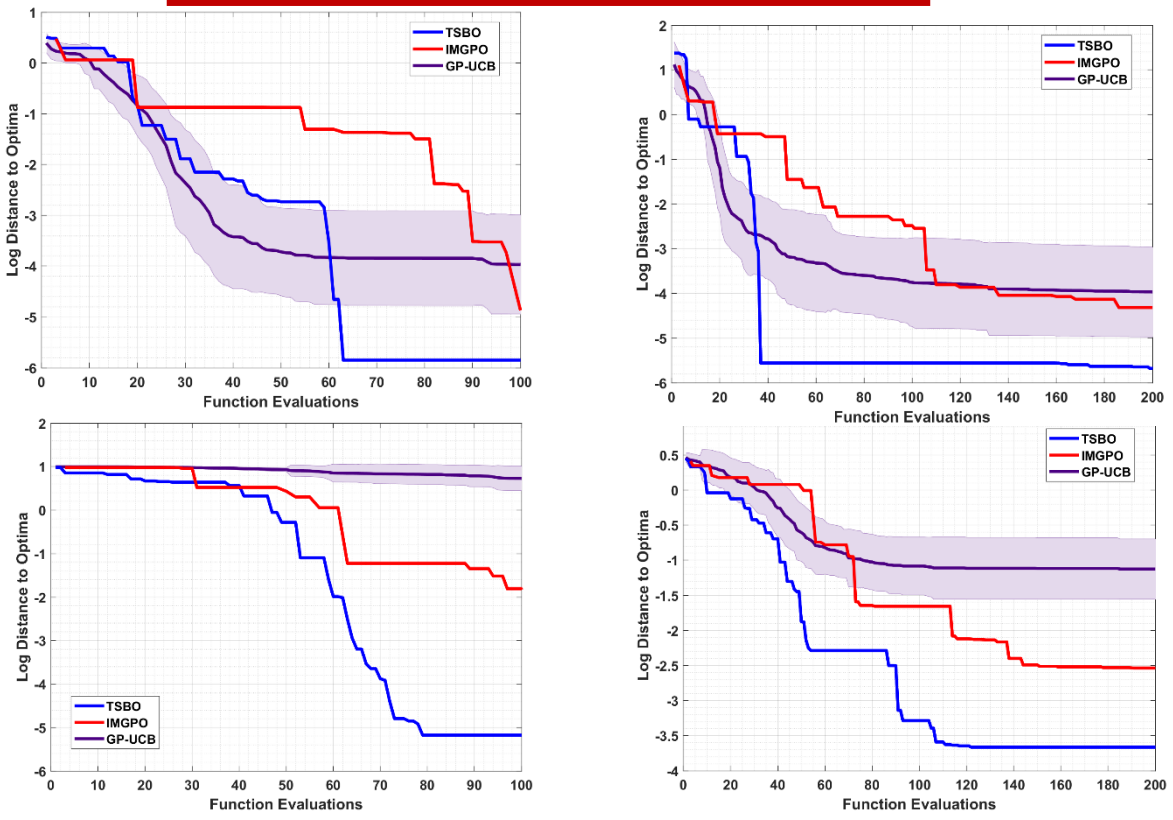
Two-stage Bayesian Optimization



- First, Fast Exploration; Second, Pure Exploitation
  - Coarse and fine tuning
- Hierarchical Partitioning Scheme
  - Reduces number of simulations required
- Active learning of acquisition function

TSBO: A CAEML Innovation (Not the Fastest Algorithm, but …)

Fewer iterations to convergence



Note: IMGPO is a known algorithm that is publicly downloadable

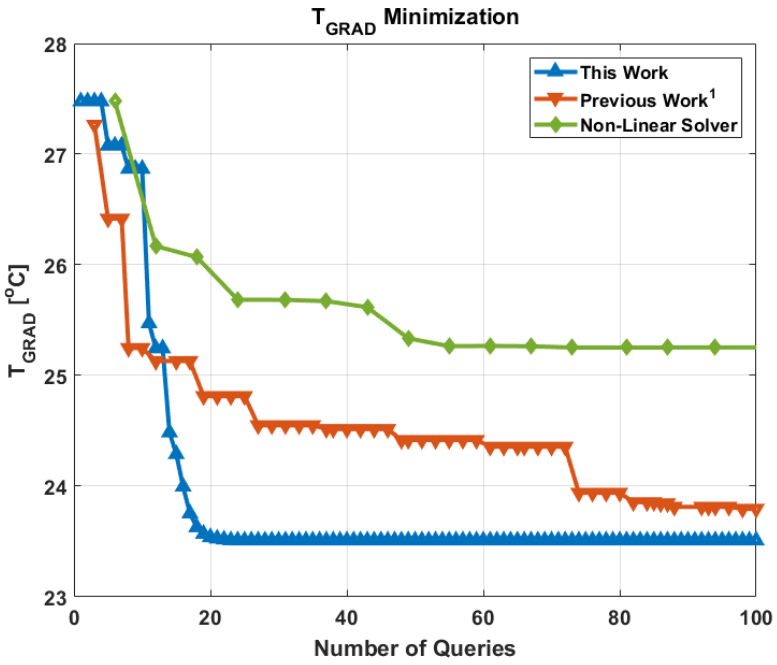
Algorithm Run Times for 100 iterations

	2D BRANIN	3D HARTMANN	4D SHEKEL	6D HARTMANN
TSBO	6.15s	4.35s	6.79s	6.51s
IMGPO	1.28s	1.35s	1.83s	1.91s
GP-UCB	10.4s	10.5s	11.4s	12.2s

- With the cost of few seconds of algorithm run time, TSBO reduces number of simulations required for accurate optimization

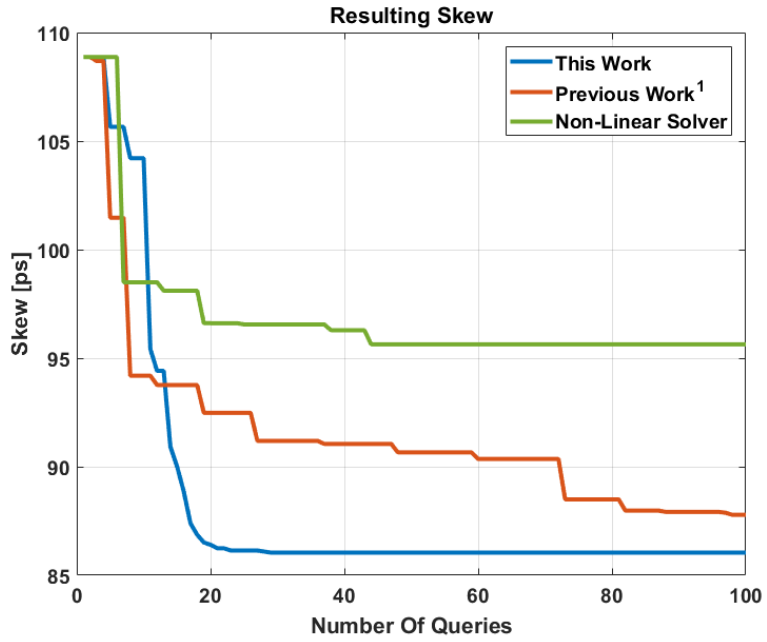
◆**For EDA applications in which simulations are expensive, minimizing the number of physical simulations is the goal**

Results: Clock Skew Minimization



~4X Faster

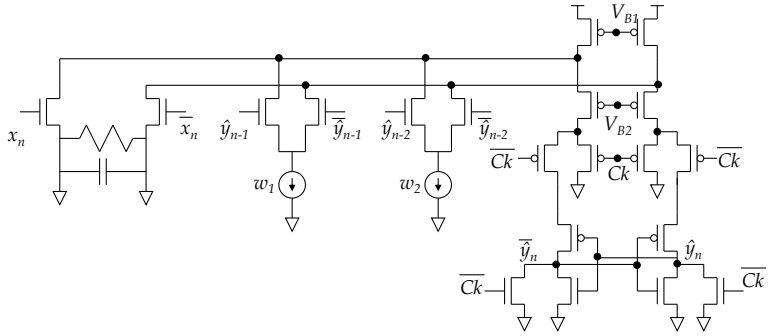
S. J. Park et al., Trans. VLSI, 2017.

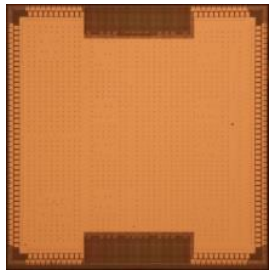


	Non Linear Solver	Previous Work[1]	This Work
T_{GRAD} [°C]	25.2 (+%9.4)	23.8 (+%4.7)	**23.5**
Skew [ps]	96.6 (+%12.3)	88 .0 (+%2.3)	**86.0**
CPU Time (Normalized) *	3.96	3.76	**1.00**

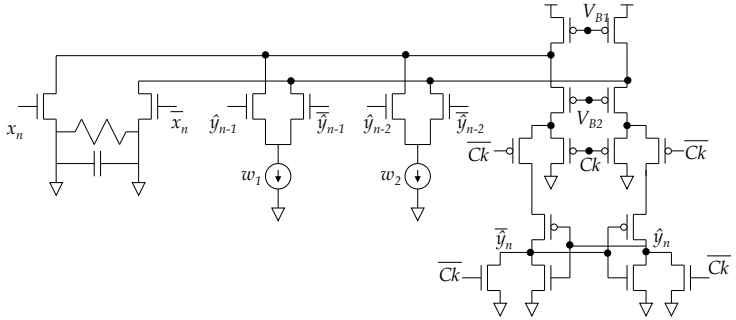
IP Reuse: Problem Statement

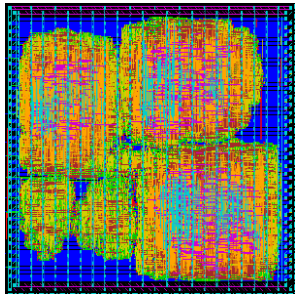
- **Port analog, RF, and custom digital IP from one technology to another**





Foundry X
Node A
CMOS/BiCMOS

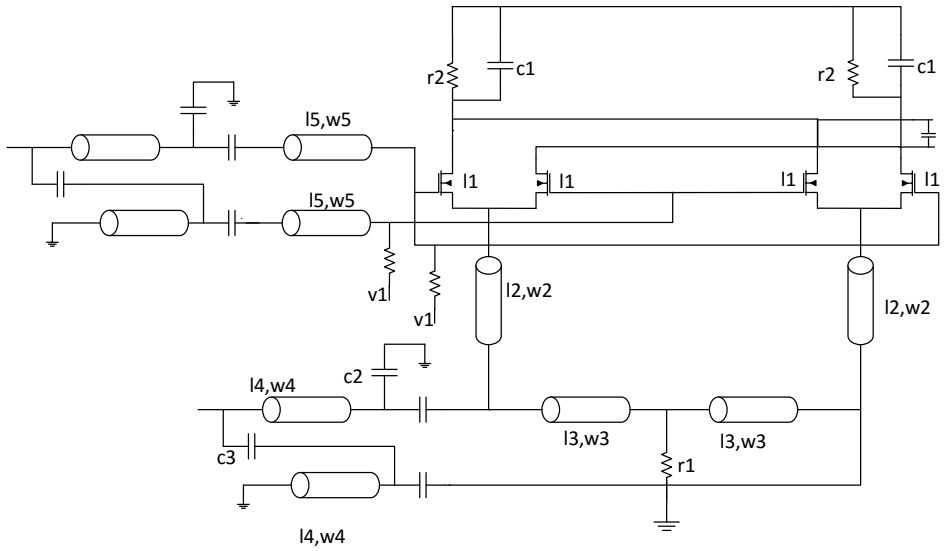




Foundry Y
Node B
SOI

Example: 130nm SiGe BiCMOS to 22nm FD-SOI

79-GHz mixer



20 dimensions

Physical Design Parameters
X=l1,l2,l3,l4,l5,w2,w3,w4,w5,r1,r2,c1,c2,v1
Target Electrical Parameters
Y1 = Voltage Gain
Y2 = Noise Figure
Y3 = LO_RF_Feedthrough
Y4 = S11(<-5dB)
Y5 = S22(<-3dB)
Y6 = iP1dB

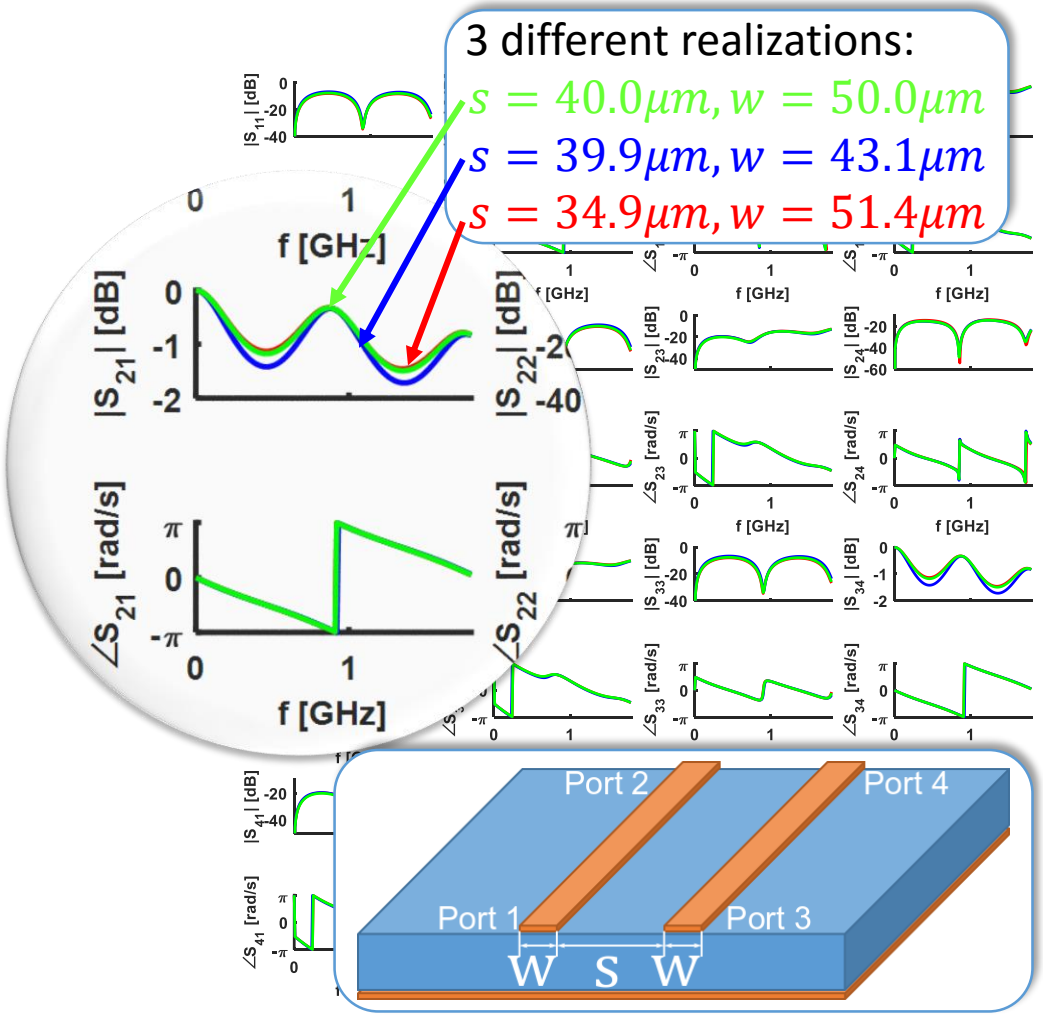
	22nm	**130nm**	**130nm manual design**
Voltage Gain	7.04	22.8	18.285
Noise Figure	16.138	10.92	9.39
LO_RF_Feedthrough	-36.8448	-34.54	-27.59
S11	-6.8	-6.1	not met -4.6
S22	-4.79	-3.13	-3.3
IP1dB	-0.694	-8.26	-4.67

Applications of ML to EDA

- Design Optimization
  - Clock skew minimization for 3D-IC
  - IP Reuse
- Generative Modeling
  - Stochastic models of PCB interconnects
  - PPA prediction
- Circuit Macro-modeling
  - Recurrent neural network
- Data Analytics
  - Early detection of hardware failure
- Additional examples from CAEML

Interconnect Modeling

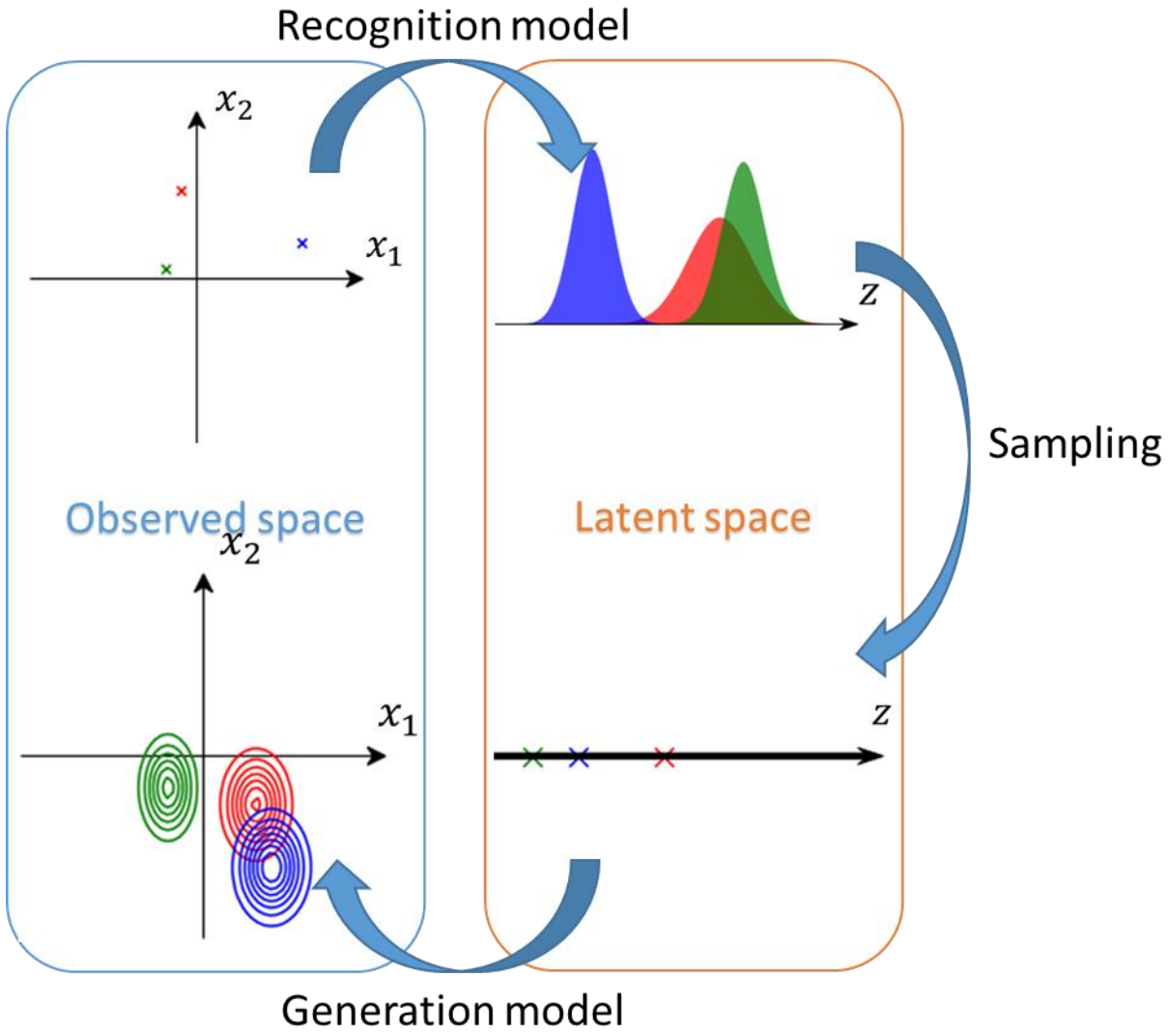
- Scattering parameters characterize the electrical behavior of circuit interconnects
  - Complex, matrix-valued function of frequency
  - Stored as frequency-tabulated matrices at discrete frequency points within band of interest
  - Can be used as generic multiport network block in circuit simulation
  - Dense frequency sampling and multiport interconnects produce high-dimensional data
- S-parameters of manufactured interconnects exhibit variability
  - Manufacturing process introduces variability in material and geometric properties
  - Deviation of those properties from design value alters electrical behavior
  - Probabilistic model is necessary to capture the variability



Overlaid scattering parameters

Variational Autoencoder (VAE)

- Explain variation of observed data with latent variables and non-linear mapping
  - High-dimensional data often concentrates near low-dimensional manifold
  - Latent space is low-dimensional and easy to model
- Recognition model and generation model
  - Probability distributions parameterized by outputs of neural network
- Optimize variational bound on combined model
  - Identify the most appropriate latent space
  - Maximize reconstruction accuracy
- Advantages
  - High memory efficiency
    - Parametric statistical model offers bounded model complexity
    - Iterative training algorithm
  - Improved accuracy
    - No additional dimensionality-reduction step needed



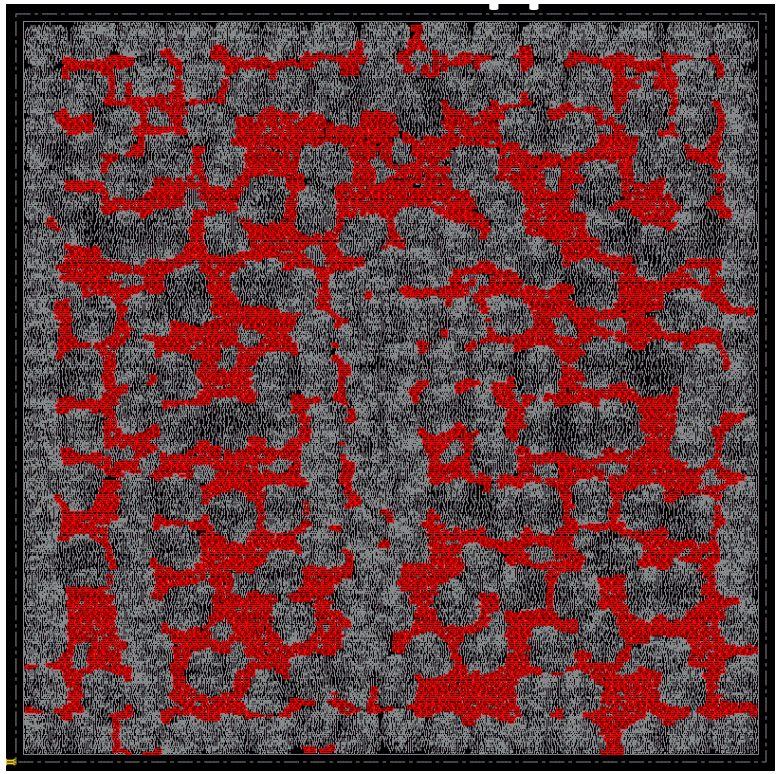
Physical Consistency

- Stability, causality, and passivity
  - Fundamental properties that interconnect model must satisfy
  - Violations potentially cause erroneous circuit simulation results
  - Difficult to build these constraints into the generative model
- Our solution: Vector fitting (VF) of generated scattering parameters (post-processing step)
  - Producing a rational function fit of system response (macro-model) based on frequency-tabulated data
  - Straightforward enforcement of stability and causality as well as perturbative passivity enforcement result in a physically consistent macro-model
  - Many circuit simulators already have VF functionality built-in

X. Ma, *2018 SPI*

Goal: Placement-to-Clock-Tree Prediction

- Numbers & images going in, numbers coming out



placement (FF in red), CLK settings

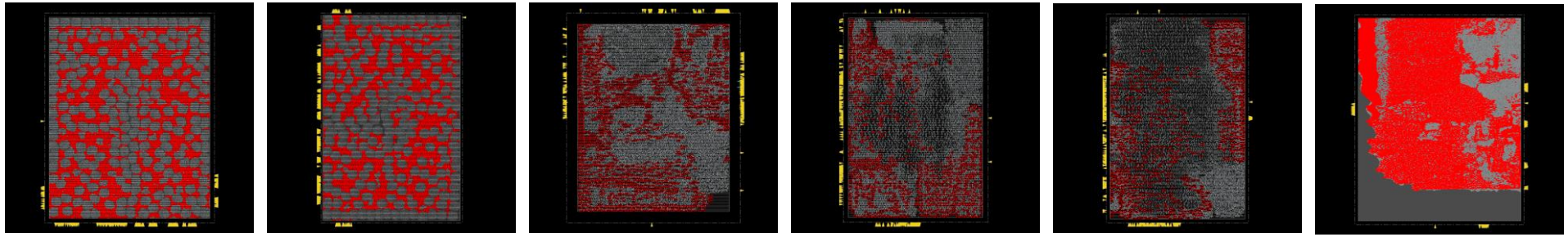
CLK WL
+
CLK skew
+
CLK power
+
best setting

clock tree quality

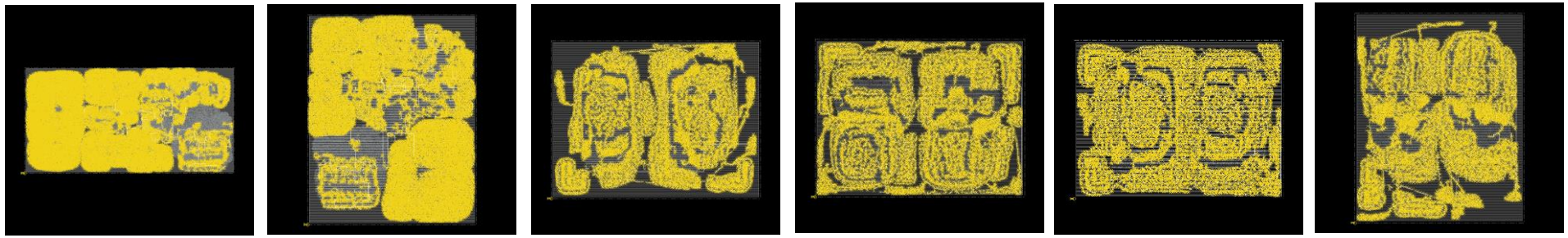
S-K Lim, GT/CAEML (to appear *ICCAD'19*)

Our Database: 24k Layouts

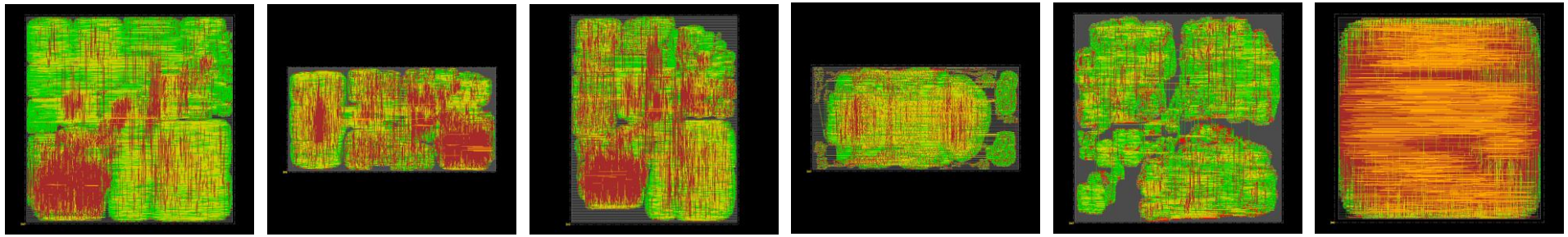
FF placement



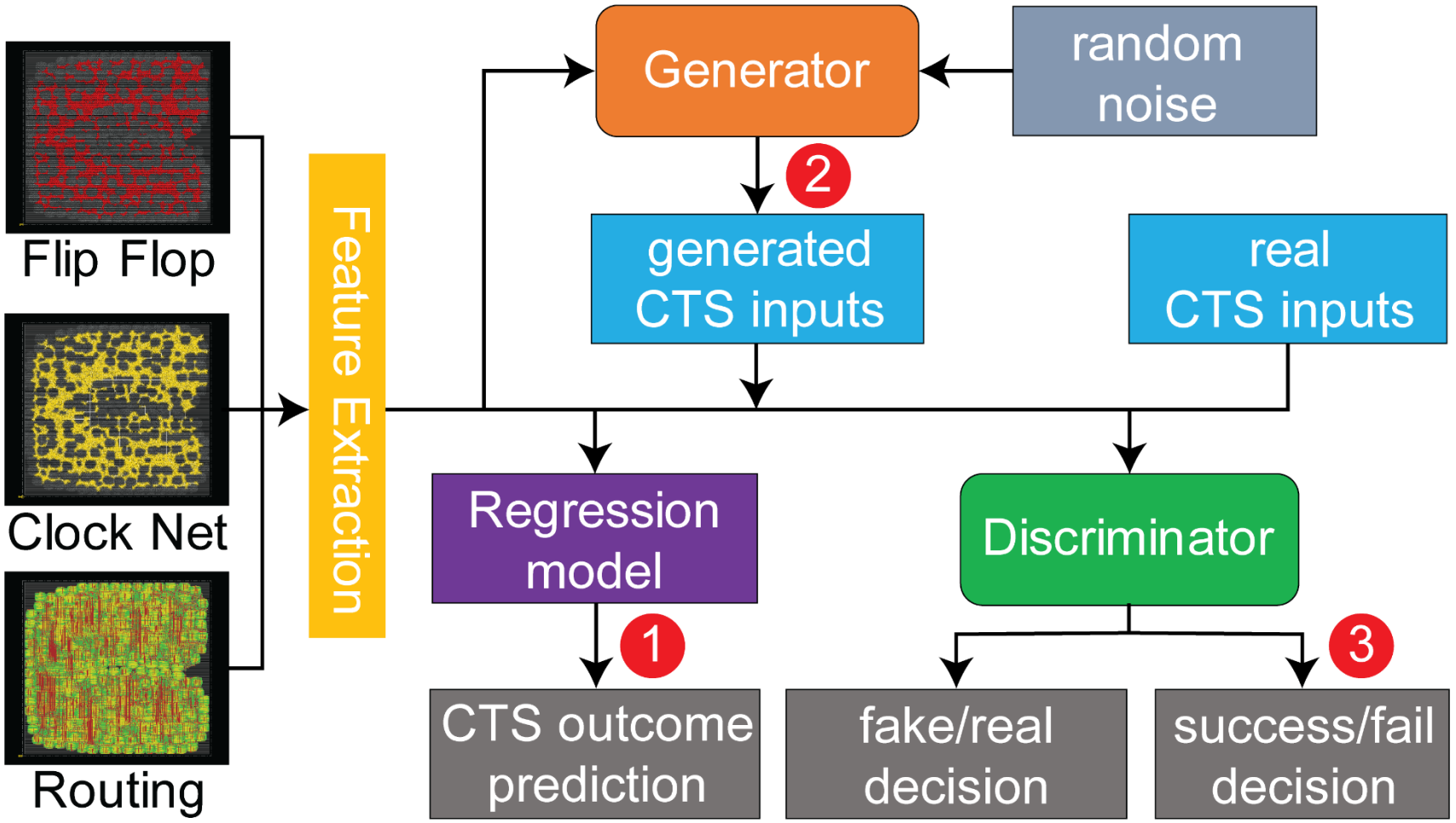
Clock nets



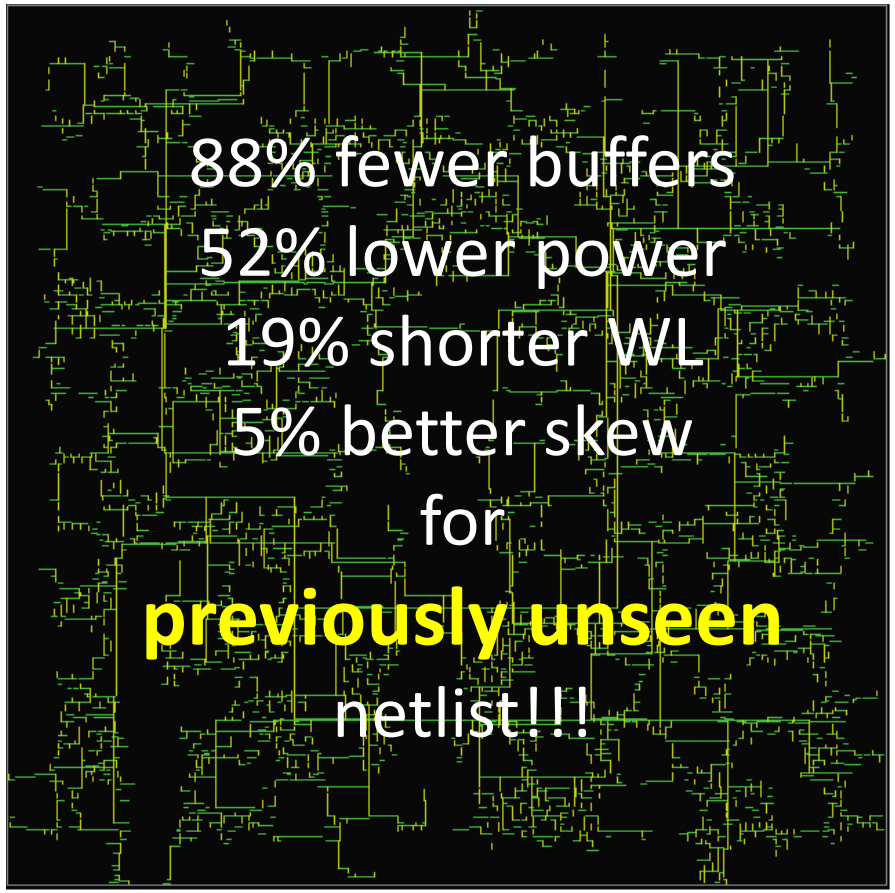
Trial routing



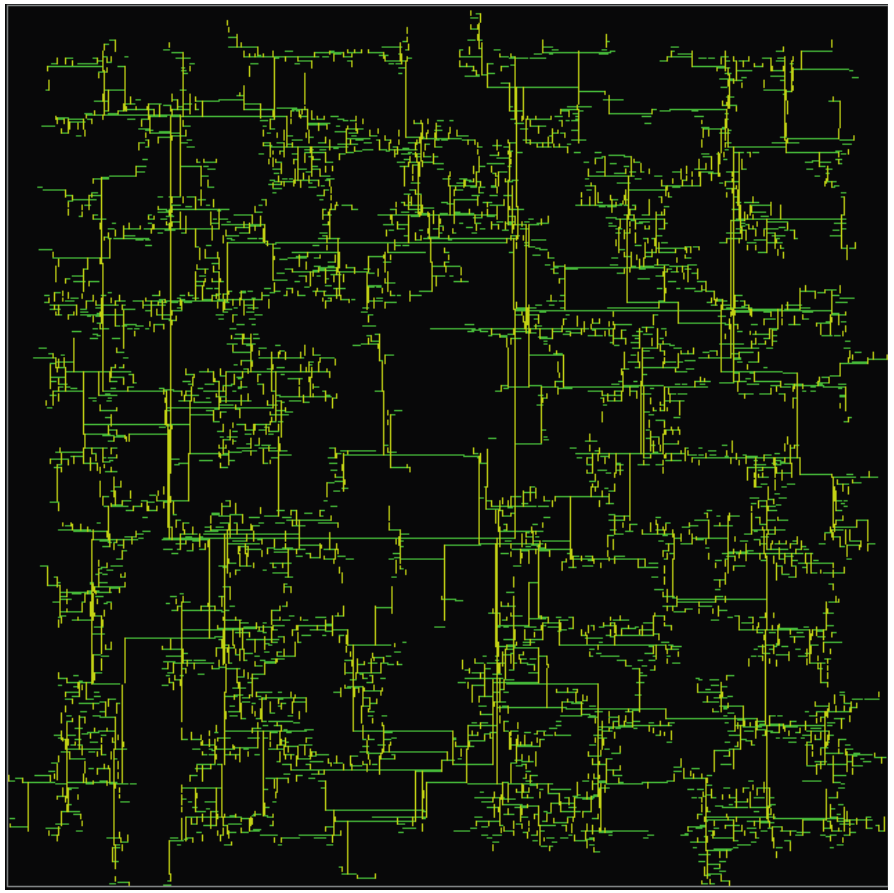
GAN-based Flow



Outperforms Commercial Auto-Setting



(a) GAN-CTS optimized



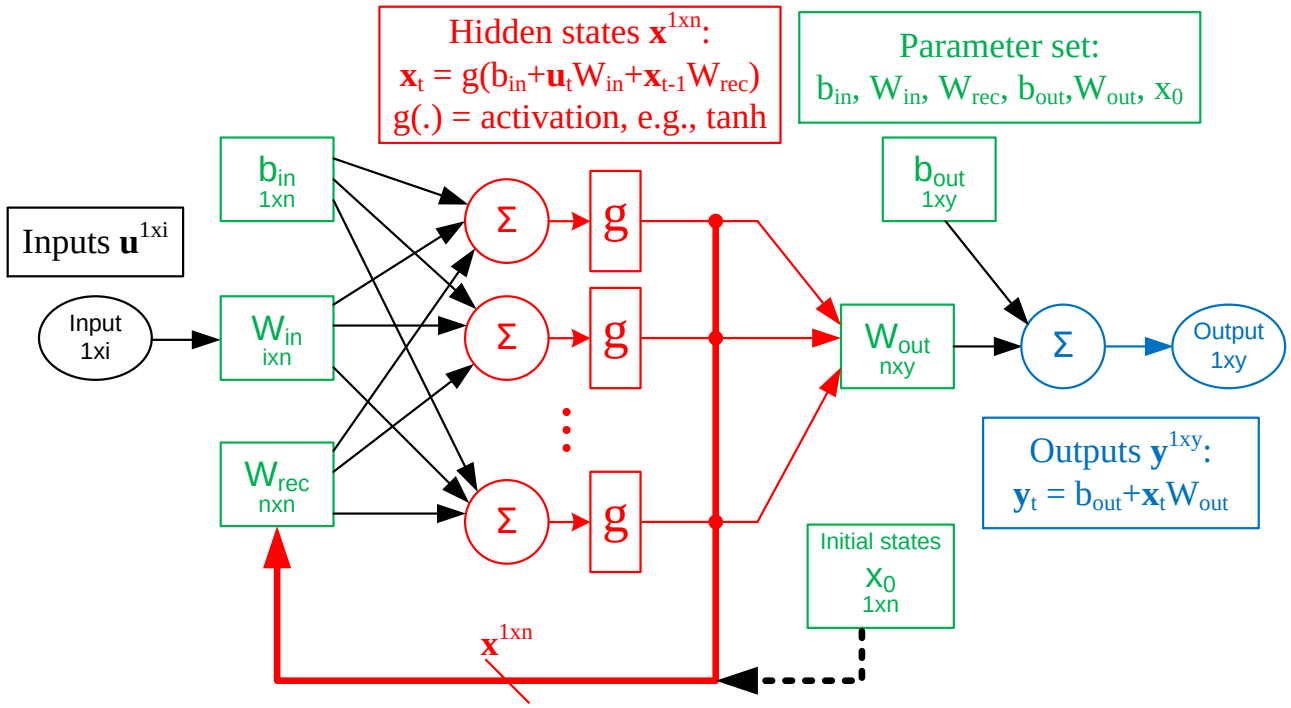
(b) commercial auto-setting

S-K Lim, GT/CAEML (to appear *ICCAD'19*)

Applications of ML to EDA

- Design Optimization
  - Clock skew minimization for 3D-IC
  - IP Reuse
- Generative Modeling
  - Stochastic models of PCB interconnects
  - PPA prediction
- Circuit Macro-modeling
  - Recurrent neural network
- Data Analytics
  - Early detection of hardware failure
- Additional examples from CAEML

Transient Modeling Using RNN



- RNN approximate any system represented by a state space model

$$\begin{cases} \dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) \\ \boldsymbol{y} = g(\boldsymbol{x}, \boldsymbol{u}) \end{cases}$$; *u* is input, *y* is output, *x* is state

- Many circuits and devices can be represented by state space models

RNN for Circuit Simulation

- "Vanilla" RNN equations

$$\boldsymbol{x}_i = \tanh\left[\boldsymbol{b}_u + W_r \boldsymbol{x}_{i-1} + W_u \boldsymbol{u}_i\right]$$
$$\boldsymbol{y}_i = \boldsymbol{b}_y + W_y \boldsymbol{x}_i$$

  - In the context of circuit modeling, u_i and y_i are voltage or current samples obtained at the i^{th} time point.
- Training data have constant time step "h." Circuit simulators use variable time step.
- Our solution: Transform discrete time RNN model to continuous time
- Implement resulting differential equation in Verilog-A
  - Differential Equation:

$$\mathrm{h} \cdot \dot{\boldsymbol{x}} + \boldsymbol{x} = \tanh(\mathrm{W_r}\boldsymbol{x} + \mathrm{W_u}\boldsymbol{u} + \boldsymbol{b}_\mathrm{u})$$

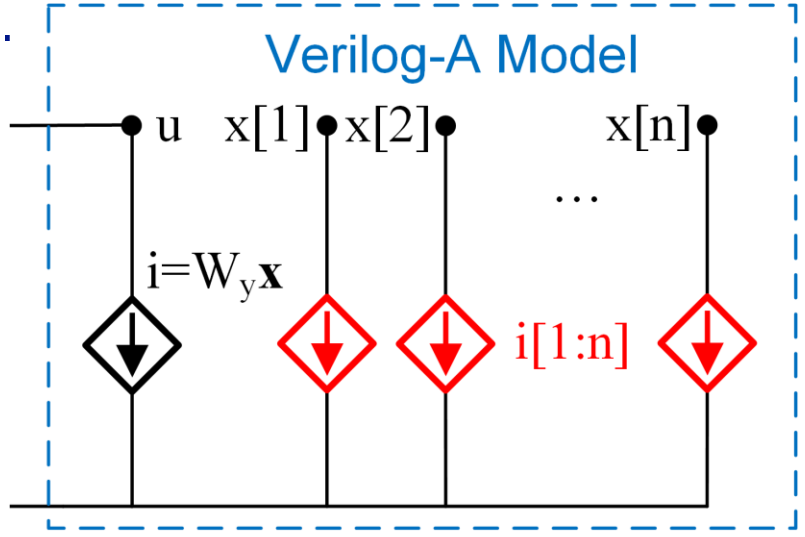
    - Create current sources

$$i[1:n] = LHS(x) - RHS(x,u)$$

    - Due to KCL, simulator solves for

$$i[1:n] = 0$$

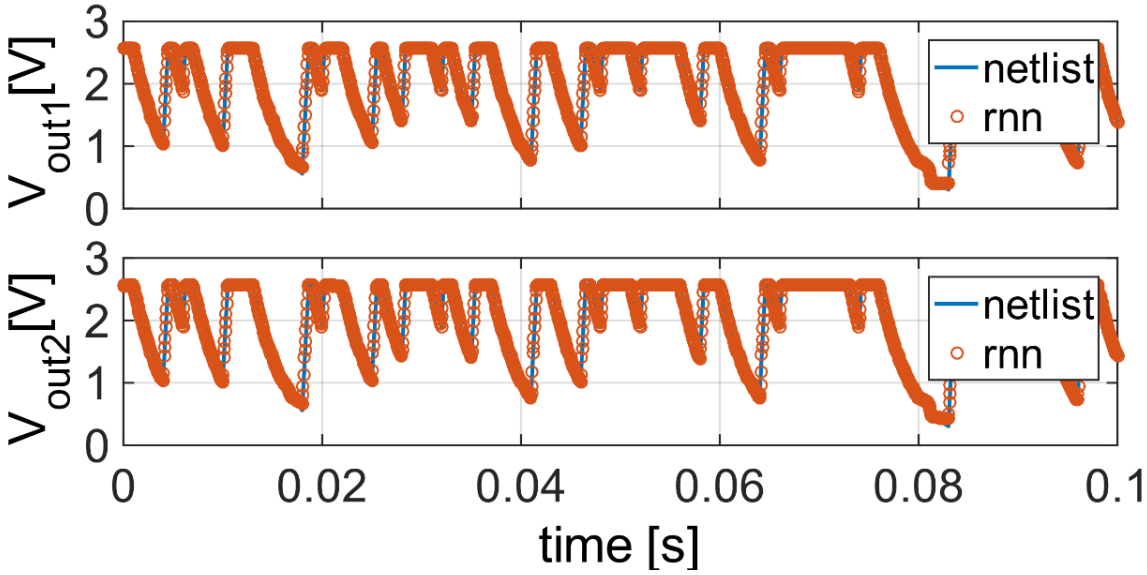
    - Hidden states $\boldsymbol{x}$ are node voltages



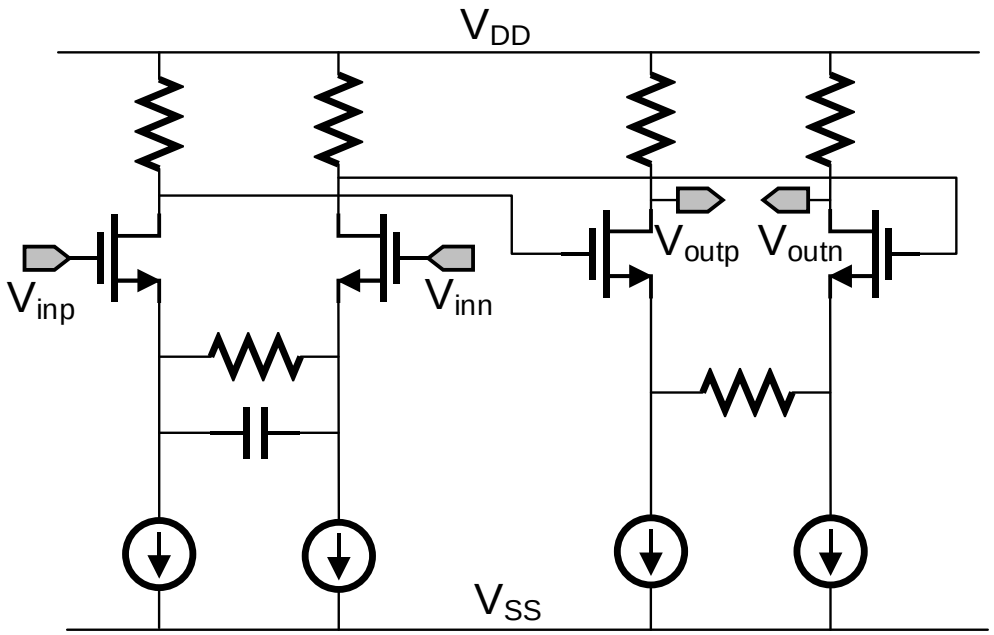
Z. Chen et al., *2017 EPEPS*

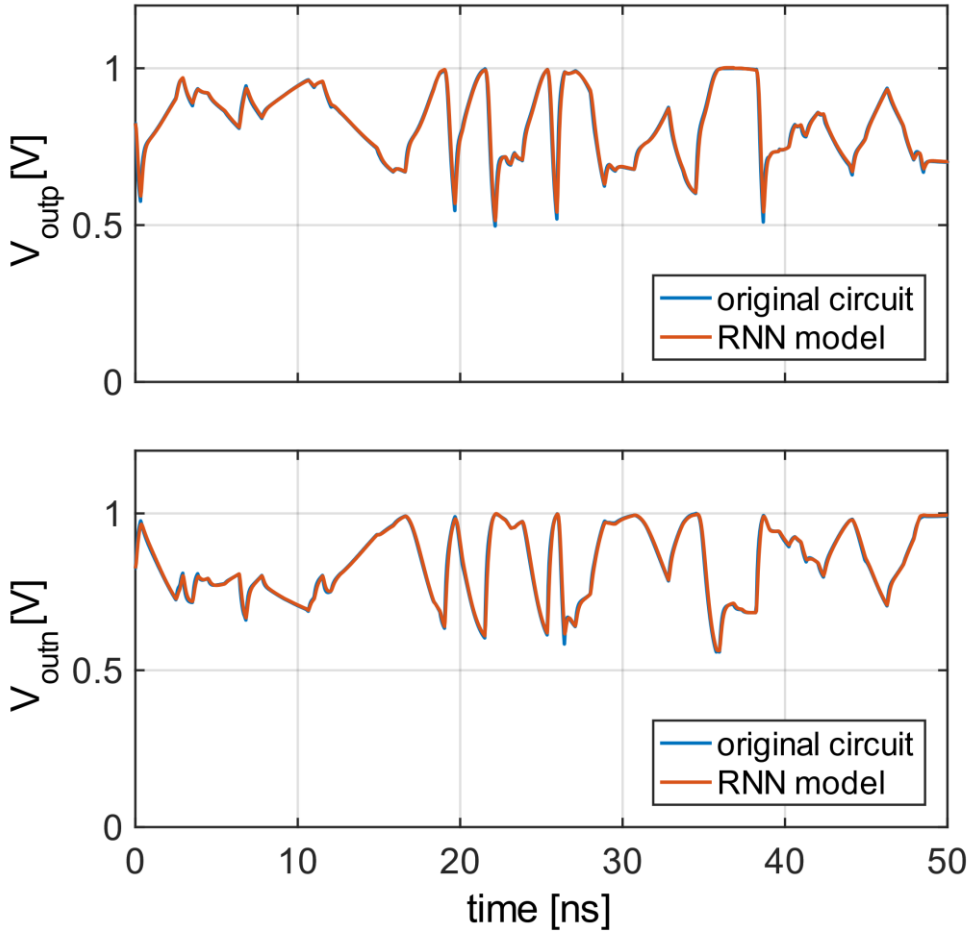
RNN Are Highly Efficient

- Test case: Encrypted netlist for industry-designed circuit
  - Over 1000 transistors
  - 7 voltage inputs, 2 voltage outputs
  - Stimuli: 2-level digital input with V_H=5V, V_L=0V, T_{bit}=1ms, t_r=t_f=1ns
- Training data generation
  - Randomly assign 1 or 0 to each input for each bit (1 ms)
  - Transient circuit simulation with full netlist
- Result
  - Basic RNN model
  - Waveform shown at right
  - Model Error < 1%
  - Simulation Time
    - **Netlist 254.5 s**
    - **RNN 5.7 s**



RNN Also Suitable to Model Analog Circuits





- Continuous-time linear equalizer
- Model evaluated for previously unseen inputs

Applications of ML to EDA

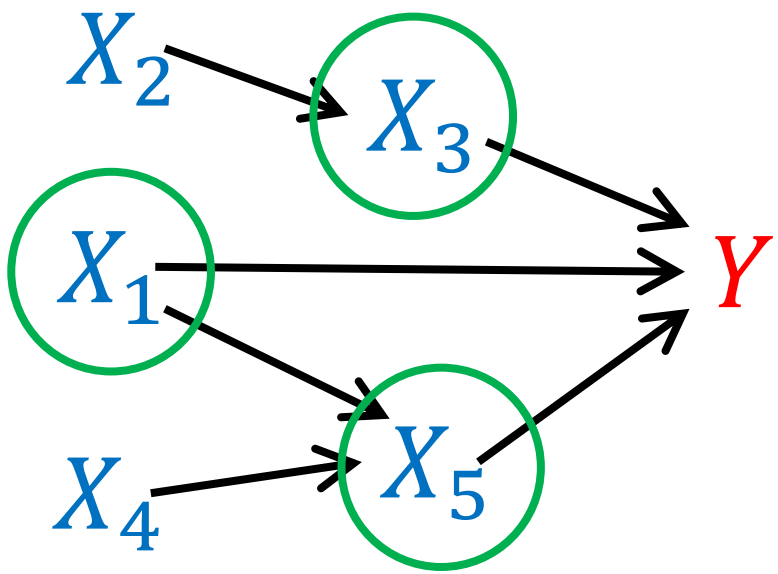
- Design Optimization
  - Clock skew minimization for 3D-IC
  - IP Reuse
- Generative Modeling
  - Stochastic models of PCB interconnects
  - PPA prediction
- Circuit Macro-modeling
  - Recurrent neural network
- Data Analytics
  - Early detection of hardware failure
- Additional examples from CAEML

Hardware Failure Prediction

- Data center HDD and SSD failure prediction
  - Model inputs (features) are SMART attributes
    - *SMART: self monitoring, analysis and reporting technology*
- Maximize FDR (failure detection rate) subject to requirement that FAR < 1% (false alarm rate)
- Failure must be predicted at least one day in advance for the information to be actionable
- Perform feature selection
  - Of the available model inputs, select the minimal set of prognostic ones (e.g. eliminate redundant data)

Causal Feature Selection

Objective: Identify the direct causes of Y, represented as the parents of Y; those features will be inputs to the model



Note that we do not try to find the whole structure.

Proposition: Let Ω be the set of all features.
X_i is not a direct cause of Y if it is independent of Y conditioned on $(\Omega - X)$.

Calculating Conditional Independence

We use *Conditional Mutual Information* (CMI) to evaluate conditional independence

$$I(X;Y|Z) = D_{KL}(P_{X,Y|Z} \| P_{X|Z} P_{Y|Z})$$

$$= \sum_{x,y,z} P(Z=z) P(X=x, Y=y|Z=z) \log \frac{P(X=x, Y=y|Z=z)}{P(X=x|Z=z) P(Y=y|Z=z)}.$$

Proposition: $X \perp Y \mid Z$ if and only if $I(X;Y|Z) = 0$.

Calculating Conditional Independence

We use *Conditional Mutual Information* (CMI) to evaluate conditional independence

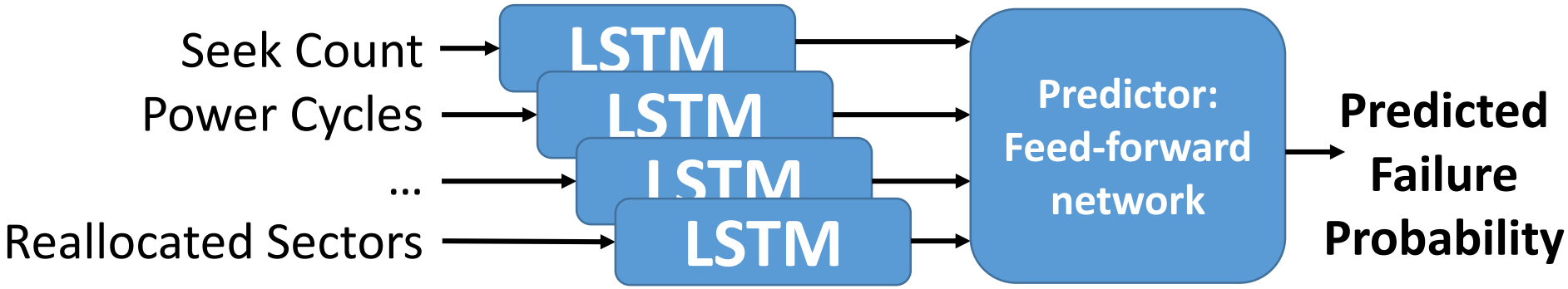
$I($ ... $z)$.

Practical Issue: Estimating mutual information in a high-dimensional setup is challenging.

.

Proposition: $X \perp Y \mid Z$ if and only if $I(X;Y|Z) = 0$.

Neural Network Model Structure

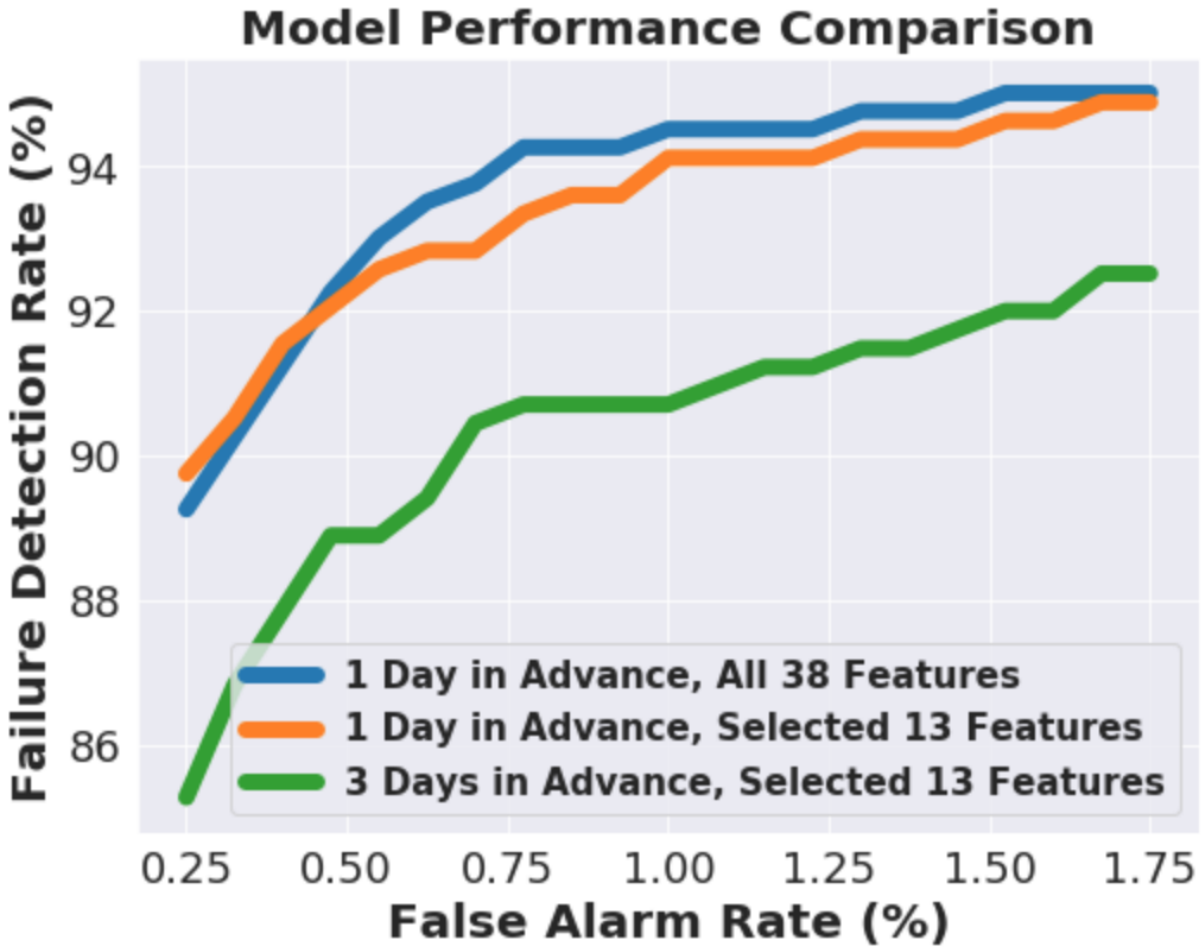


HDD Failure Prediction

Prognostic HDD diagnostics:

1. **Reported Uncorrectable Errors**
2. **SATA Downshift Errors**
3. **Reallocated Sectors**
4. **Current Pending Sectors**
5. **Load Cycles**
6. **Head Flying Hours**
7. **Power On Hours**
8. **Seek Count**
9. **Start/Stop Count**
10. **Power Off Retracts**
11. **Power Cycles**
12. **Total Logical Blocks Written**
13. **Total Logical Blocks Read**

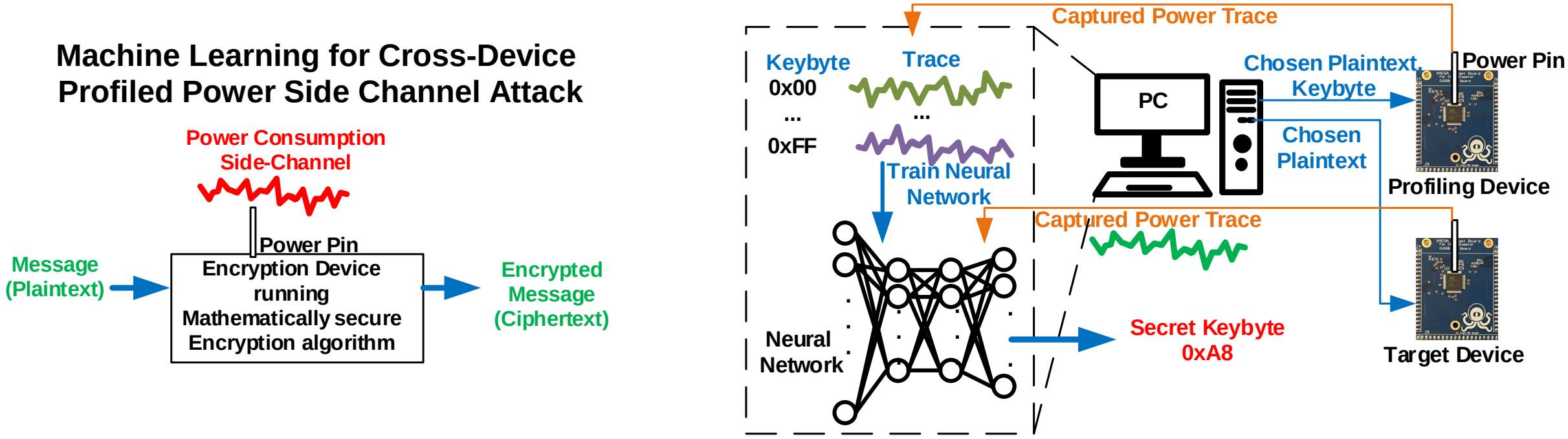
SMART data from BackBlaze



Applications of ML to EDA

- Design Optimization
  - Clock skew minimization for 3D-IC
  - IP Reuse
- Generative Modeling
  - Stochastic models of PCB interconnects
  - PPA prediction
- Circuit Macro-modeling
  - Recurrent neural network
- Data Analytics
  - Early detection of hardware failure
- Additional examples from CAEML
  - Side channel attacks
  - Back-end IC design
  - Signal integrity

Side-channel Attacks on iOT Devices

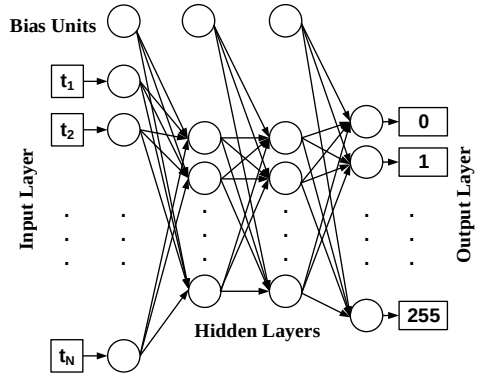


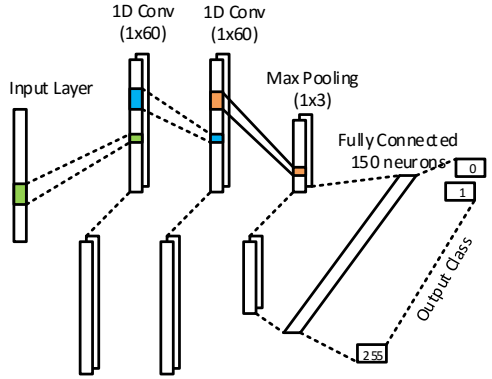
- Power Consumption of an encryption device reveals information about secret key (called **Power Side-Channel Leakage**).
- **Neural Networks** can be trained on the **captured power traces** obtained from a **Profiling device** in control of an adversary and later used to reveal **secret key** of a **Target device** from its captured power trace.

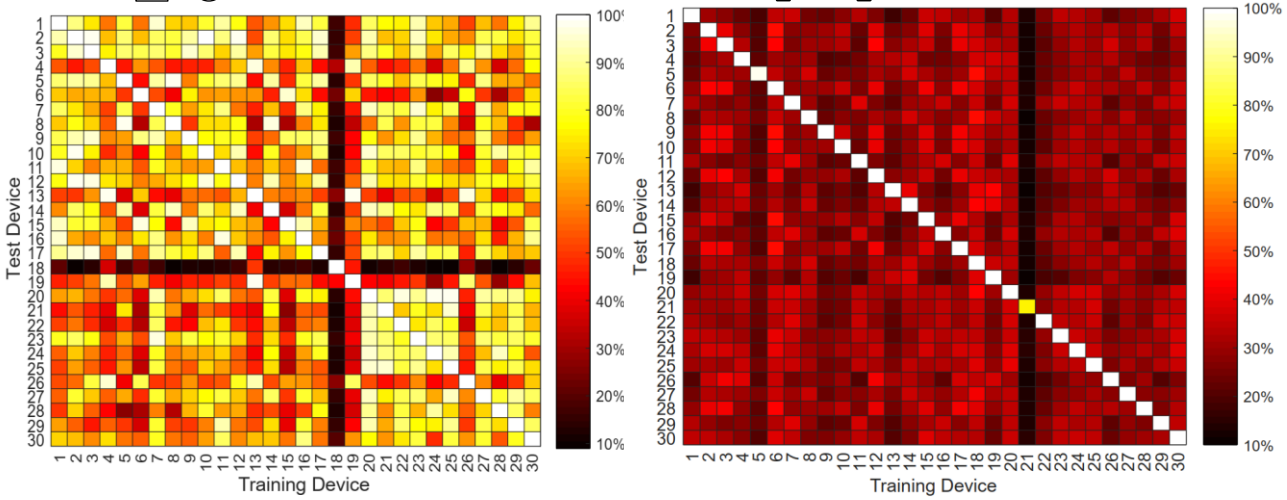
Machine Learning for Trusted Platform Design

A. Raychowdhury, GT/CAEML

ML Methods Employed (State of the art)



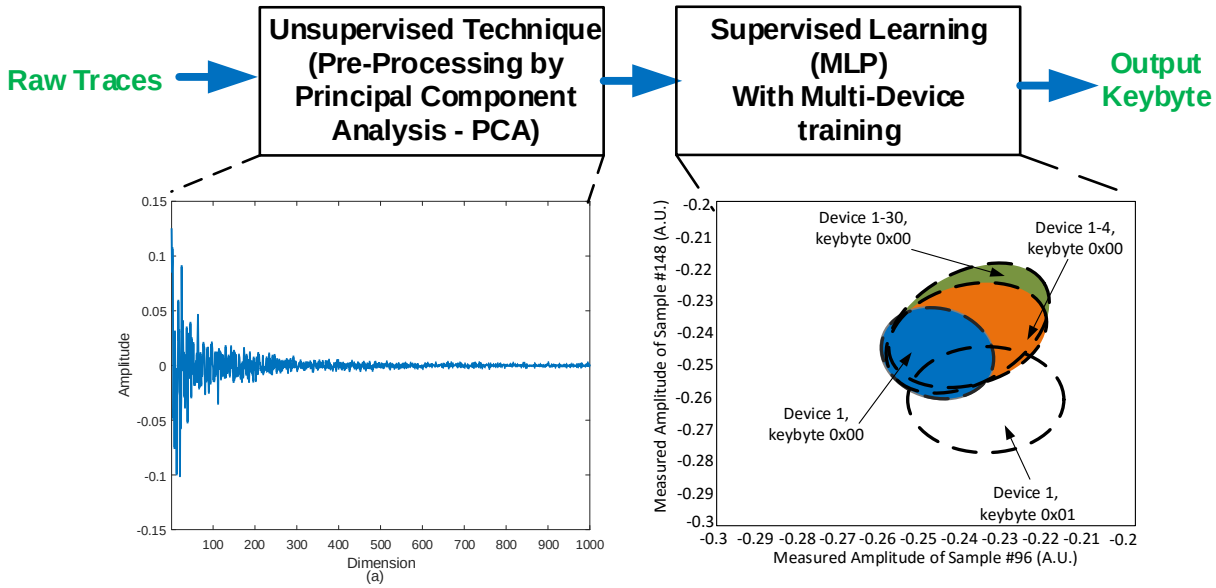




(a) Multi-Layer Perceptron (MLP) (b) Convolutional Neural Network (CNN)

Shortcomings: Overfitting to device Specific Leakage, Poor Cross-Device Attack Accuracy

Proposed ML Method (PCA-MLP)



Main Contribution: Projection of traces to Principal Subspaces and Using Multiple device to mitigate device-to-device variations

Results

Benefits over the current State-of-the-Art

Number of Training Devices	*Test Accuracy (%)*								
	MLP			*PCA – MLP*			*CNN*		
	Average	Maximum	Minimum	Average	Maximum	Minimum	Average	Maximum	Minimum
1	61.98	98.70	2.95	90.09	99.94	53.18	29.97	44.86	10.09
2	79.14	99.92	4.47	96.65	**99.99**	71.28	47.75	74.42	21.27
3	90.76	99.93	8.93	99.37	**99.99**	**90.82**	78.69	98.93	51.15
4	91.72	99.95	8.02	**99.43**	**99.99**	89.21	80.39	94.63	60.08

**Does not include Test Accuracy for Devices used in Training Set*

Progressive Improvement with Multi-Device Training Compared to Single-Device Training

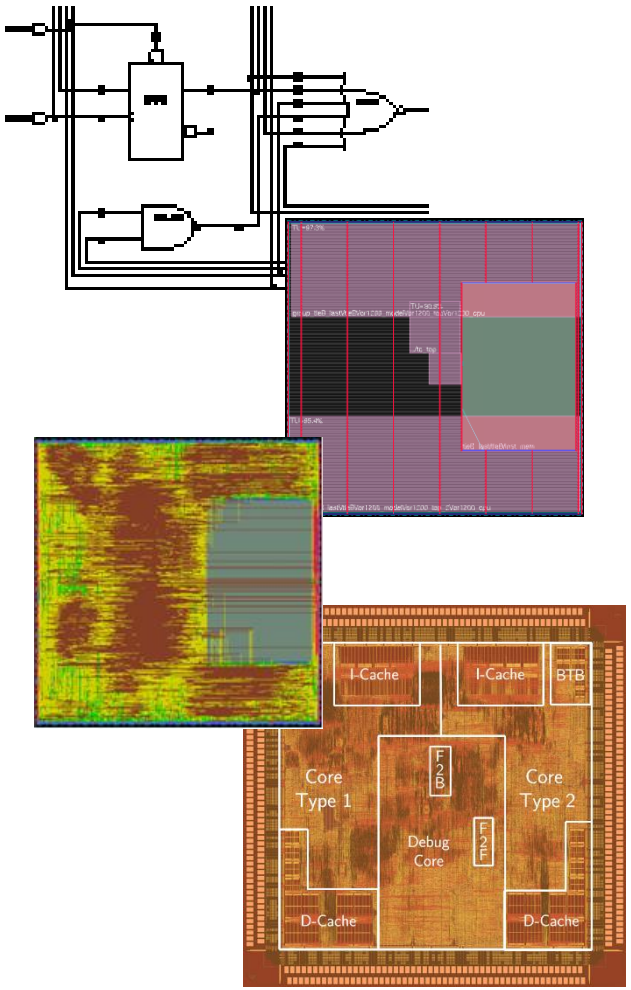
~8-20% improvement in average accuracy with PCA-MLP

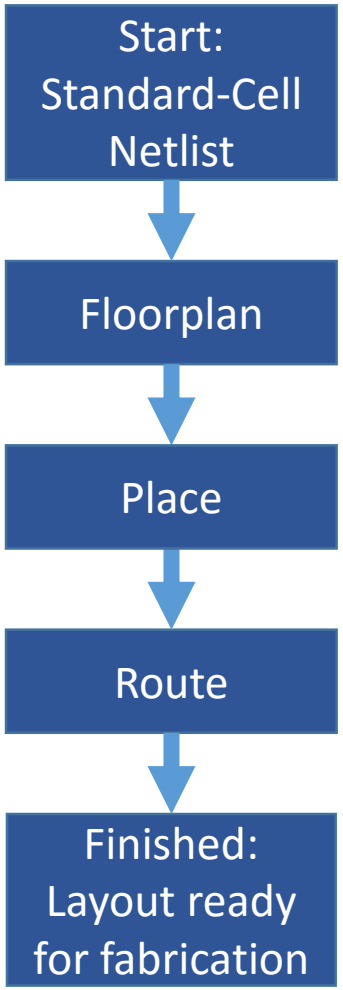
An order of magnitude improvement in minimum accuracy with PCA-MLP compared to MLP

~30% better accuracy with PCA-MLP than CNN based approach

A. Golder et al., *IEEE Trans. VLSI*, 2019.

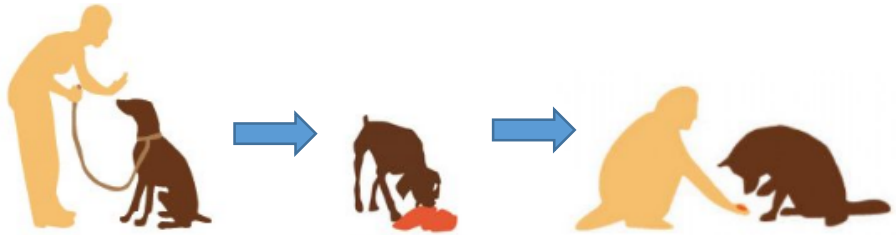
ML for Digital Integrated Circuit Design





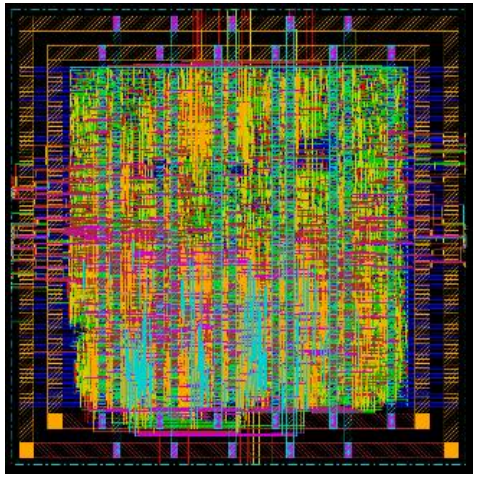
◆ML for Back-End IC Design

- Problem
  - Tools have many settings, relationship to outcomes unclear, design is slow
  - High cost of design, schedules uncertain
- Solution
  - Use Machine-Learning techniques to make sense of complex relationships
  - Train models to “fetch” the best settings
  - Back-end designers become trainers, rather than the knob-turners



Example Application

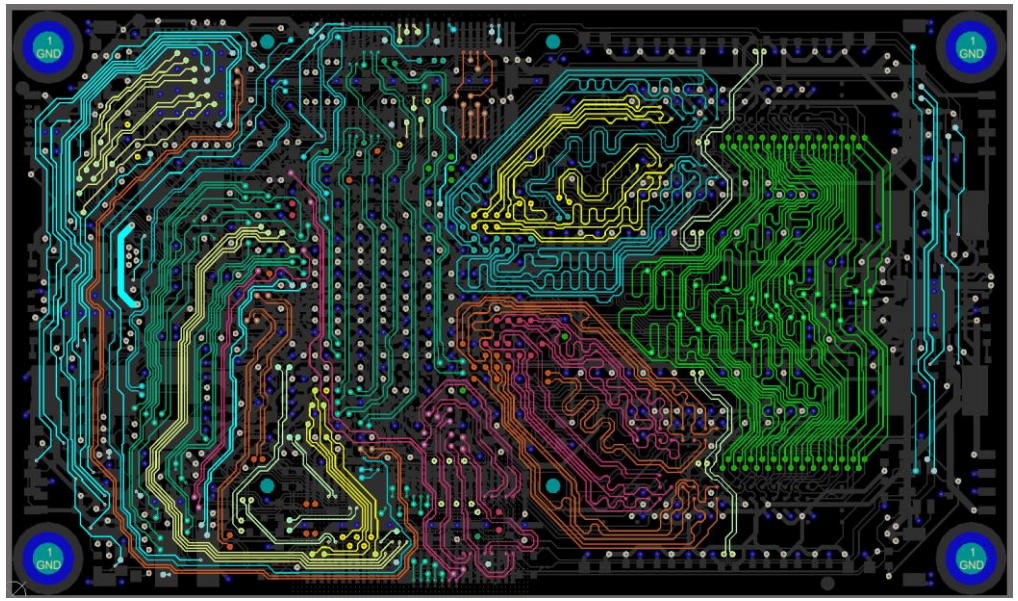
Iter	CLKper	Den.	Lyr	Max Skew	Sink Max Tran	Cong.	Viol	Hold slack	Setup Slack	Comments
1	10	0.6	8	300	400	0.28H/1.51V	105	-61.3ps	6.46ns	Over-congested; Hold time violated
2	10	0.5	8	300	400	0.03H/0.39V	6	-48.7ps	6.55ns	Over-congested; Hold time violated
3	10	0.45	8	300	400	0.02H/0.11V	0	2.4ps	6.48ns	No DRC errors; hold fixed; hold margin is low
4	10	0.45	8	200	300	0.02H/0.17V	0	10.5ps	6.37ns	Final Design



- ❑ Surrogate model provides guidance for design & optimization
- ❑ Able to achieve an optimal design with 4 iterations
- ❑ Human designer took 20 iterations

# of Standard Cells		39990
Area (μm^2)	Core	98109.284 (313.224*313.224)
	Chip	54363.008 (233.158*233.158)
Cell Density		55.4 %

Package/Board Level SI Analysis: Problem Statement



- ❑ Package/Board-level analysis is not practical due to CPU time & memory constraints.
- ❑ Current Approach to package/board design:
  - × Designer makes worst-case assumptions for pre-layout simulations
  - × Exhaustive post-layout sims are impractical

Physical design wish list:

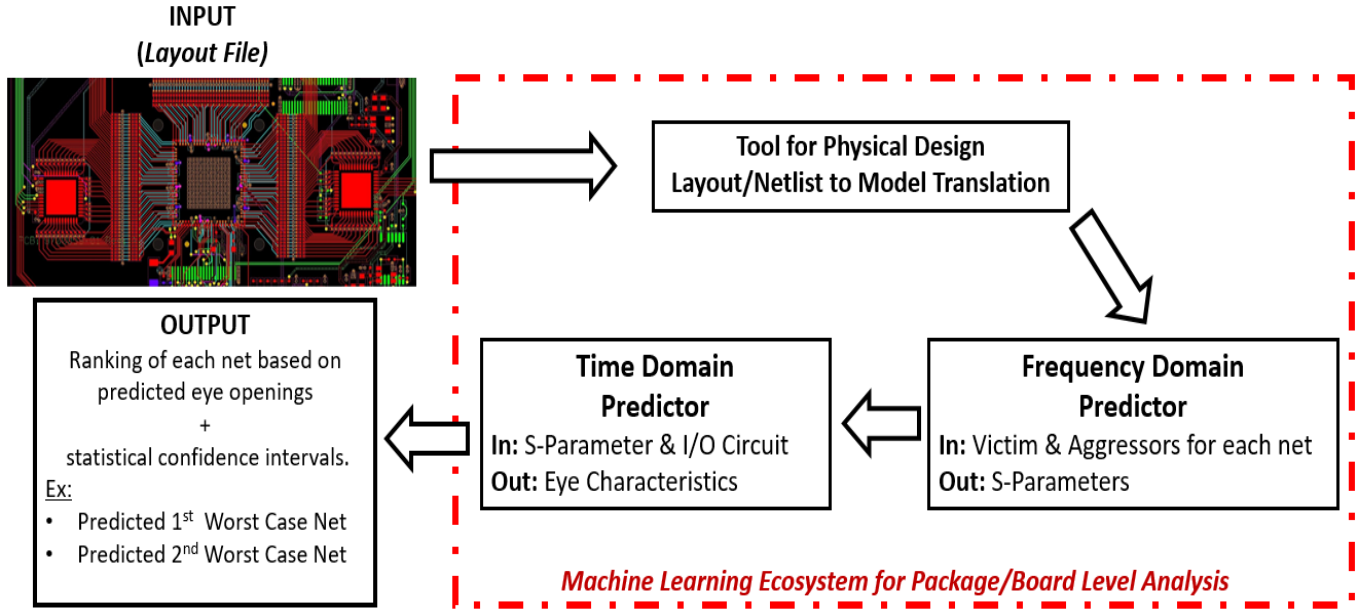
Given board file, ML method predicts worst-case performing nets w/o simulation

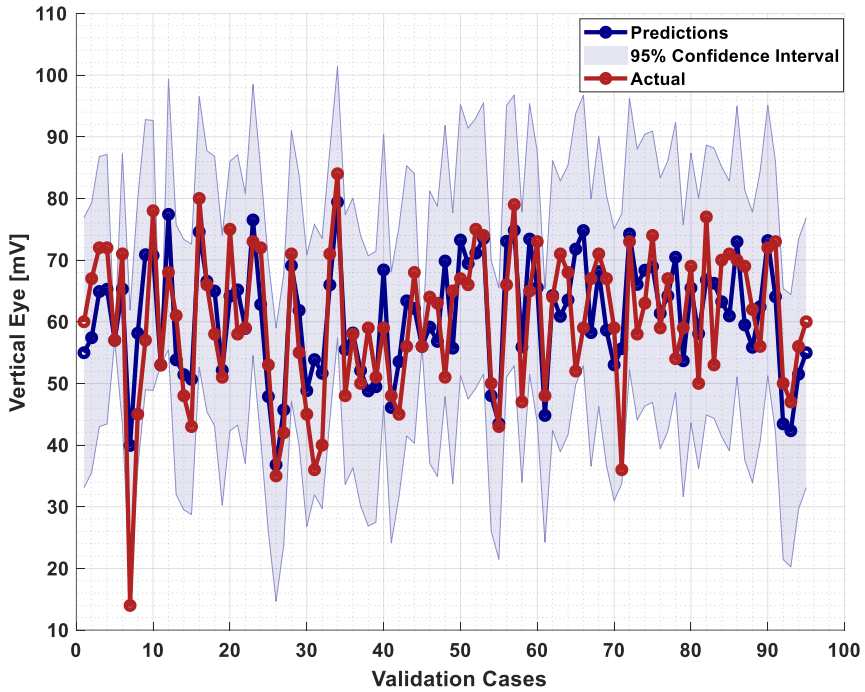
Verification effort focuses on worst-case nets

ML-generated expert knowledge for identifying “problem areas”

Uncertainty quantification: Statistical results take into account uncertainties due to manufacturing variability and lack of information about layout

Proposed Solution: Machine Learning Ecosystem



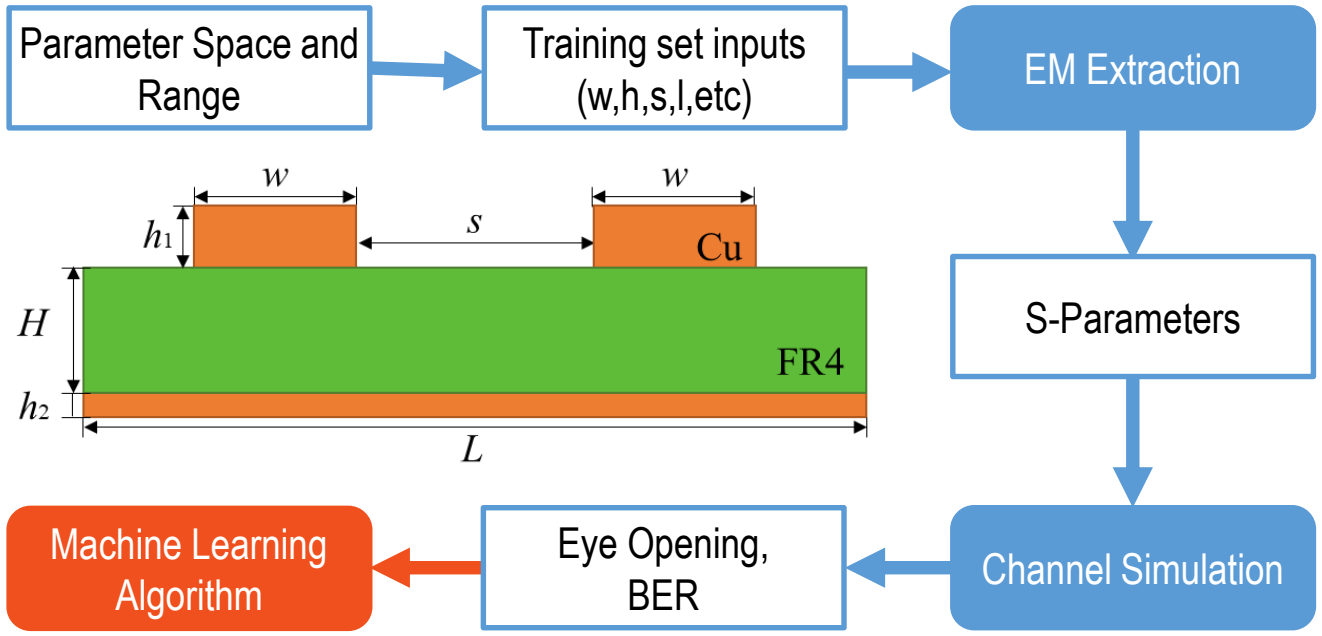


- Creating a ML ecosystem to characterize electrical performance of each net.
  - **Step 1:** Predict S-Parameters of each net based on layout (w/o EM extraction)
  - **Step 2:** Predict eye openings given S-Parameters and I/O settings (w/o simulation)
- Statistical confidence intervals to check accuracy of predictions.
- Rank nets in descending order of SI performance to determine bottleneck in the system.
- **Key Challenge to be addressed:**
  - How to make sure ML models can accurately cover every possible routing scenario?

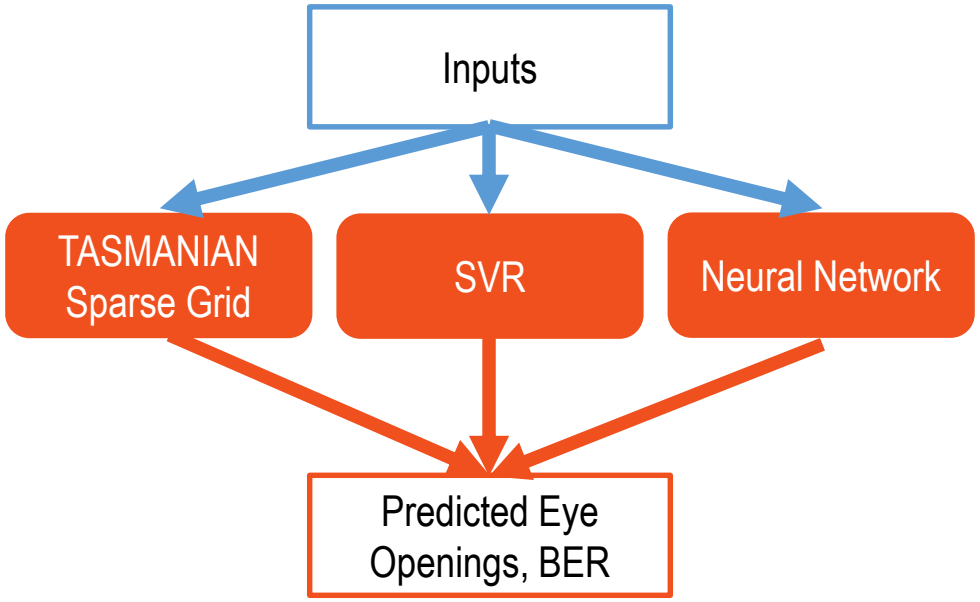
Diff pair, 16 Gb/s, 8 crosstalk aggressors
BAL-DO algorithm
H. Torun, *2018 EPEPS*

ML Surrogate Models for Eye Opening Prediction

Data Generation and Training



Prediction



TASMANIAN: Sparse Grid Interpolation

Support Vector Machine (SVM): Regression, RBF Kernel, optimal λ and C settings via GridSearchCV

Neural Network (NN): NN with backpropagation, SGD optimizer

Comparison of ML Techniques for Eye Opening Prediction

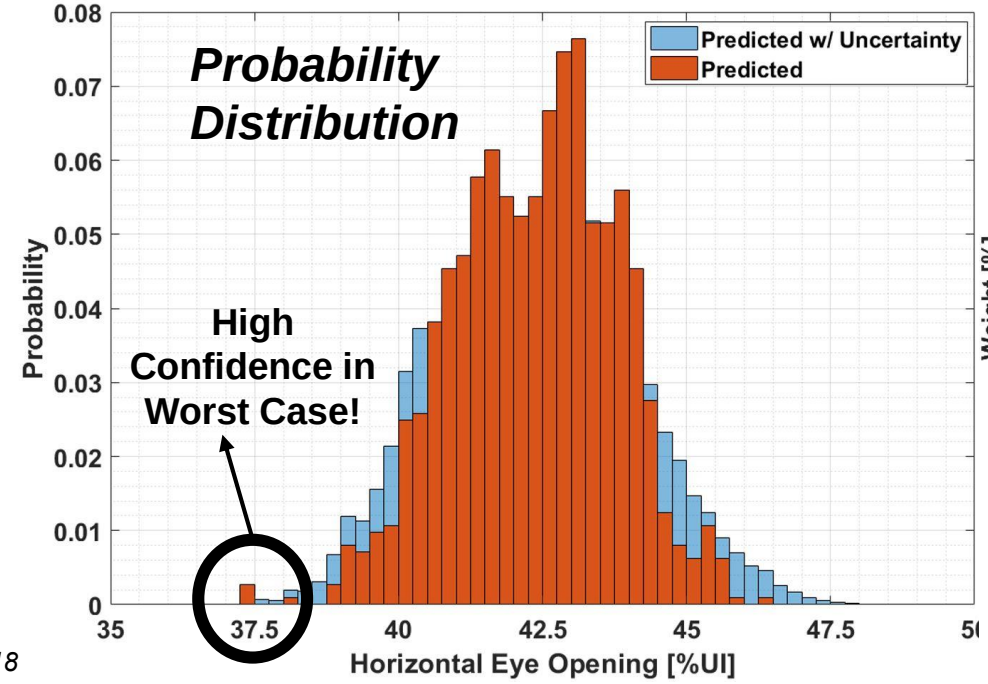
Rel. Error	Height			Width			Width At 1e-12 BER		
	TAS	SVM	NN	TAS	SVM	NN	TAS	SVM	NN
Max	0.2439	0.1108	0.1216	0.1809	0.0456	0.0784	0.1731	0.0495	0.0791
Mean	0.0382	0.0125	0.0233	0.0314	0.0105	0.0313	0.0312	0.0103	0.0297
Std	0.0374	0.0142	0.0206	0.0276	0.0091	0.0188	0.0275	0.0088	0.0176
<0.01	22.81	59.38%	29.38%	21.56%	61.56%	13.44%	23.44%	61.25%	14.06%
<0.05	74.06%	97.19%	91.56%	80.63%	100%	83.44%	80.31%	100%	88.13%
<0.1	92.19%	99.69%	99.38%	96.88%	100%	100%	97.19%	100%	100%

Training data for all 3 techniques: **721** data sets from level 7 sparse grid
Testing data: (level 8 grid) – (level 7 grid) = **320** data sets

SVM Regression provides the best results for this problem.

Eye is predicted with high accuracy without simulation.

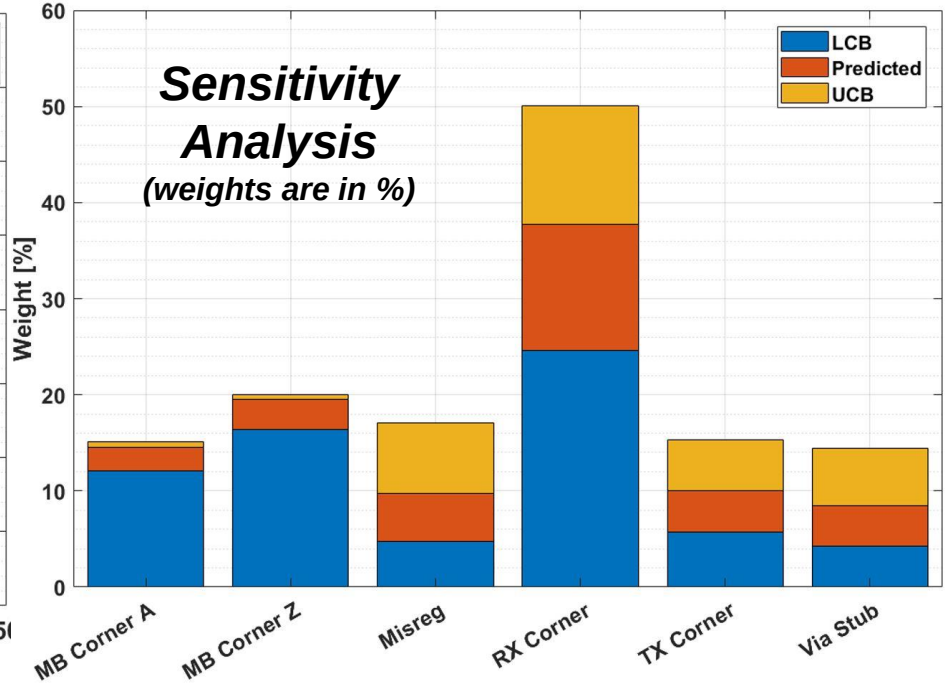
Sensitivity Analysis



H. Torun et al, EPEPS '18

$$\max\left(P\left(f^* > \tilde{f}\right)\right) = 3.01\%$$

Maximum Probability of finding a worse case



$$\max\left(E\left[f^* - \tilde{f}\right]\right) = 0.0066$$

Maximum Expected Improvement over current worst case

- ❑Instead of doing optimization to find the worst case design "corner" and then learning the function to do sensitivity analysis, BAL-DO (Bayesian active learning with drop-out) performs those tasks simultaneously
- ❑Eye PDF and sensitivity information are obtained using only 50 simulations using BAL-DO.
- ❑As the data are scarce, PDF & sensitivity analysis without uncertainty bounds can be misleading.

Acknowledgements

- Several of my colleagues in CAEML contributed slides to this presentation
  - Illinois: Xu Chen, Max Raginsky
  - Georgia Tech: Madhavan Swaminathan, Sung-Kyu Lim, Arijit Raychowdhury
  - NCSU: Paul Franzon, Rhett Davis
- Thanks to them and my other colleagues in the center for exploring how ML can help EDA
  - Plus an especially big thanks to the center's graduate students—they do all the hard work!
- The support of NSF and CAEML's industry members is gratefully acknowledged
  - Analog Devices, Cisco, Cadence, HPE, IBM, Intel, Lockheed Martin, Qualcomm, Samsung, Sandia, Synopsys, Xilinx