SYNOPSYS®
Silicon to Software™

Machine Learning-Enabled High-Frequency Low-Power Digital Design Implementation At Advanced Process Nodes

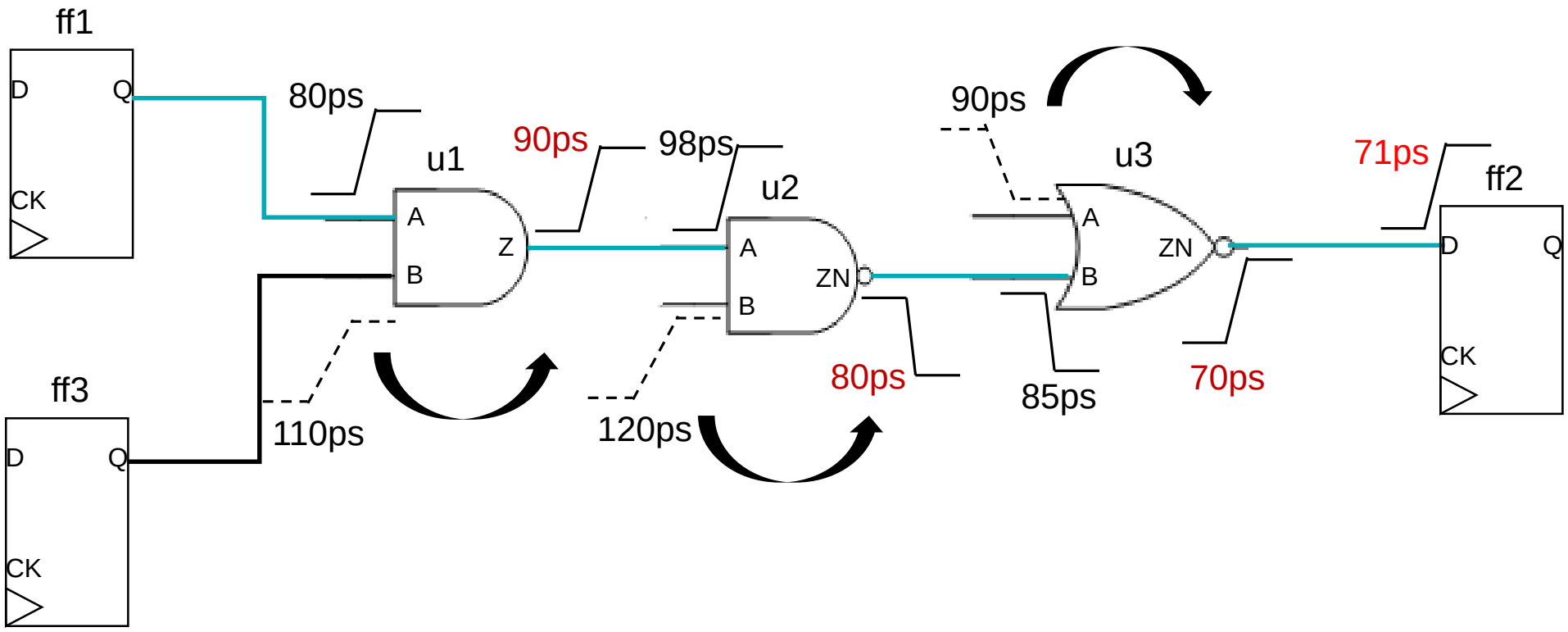
ISPD 2021

Siddhartha Nath and Vishal Khandelwal
Digital Design Group
Synopsys Inc.

Outline

- Background and Motivation
- Previous Work
- Problem Formulation
- Methodology
- Experimental Results
- Summary and Future Work

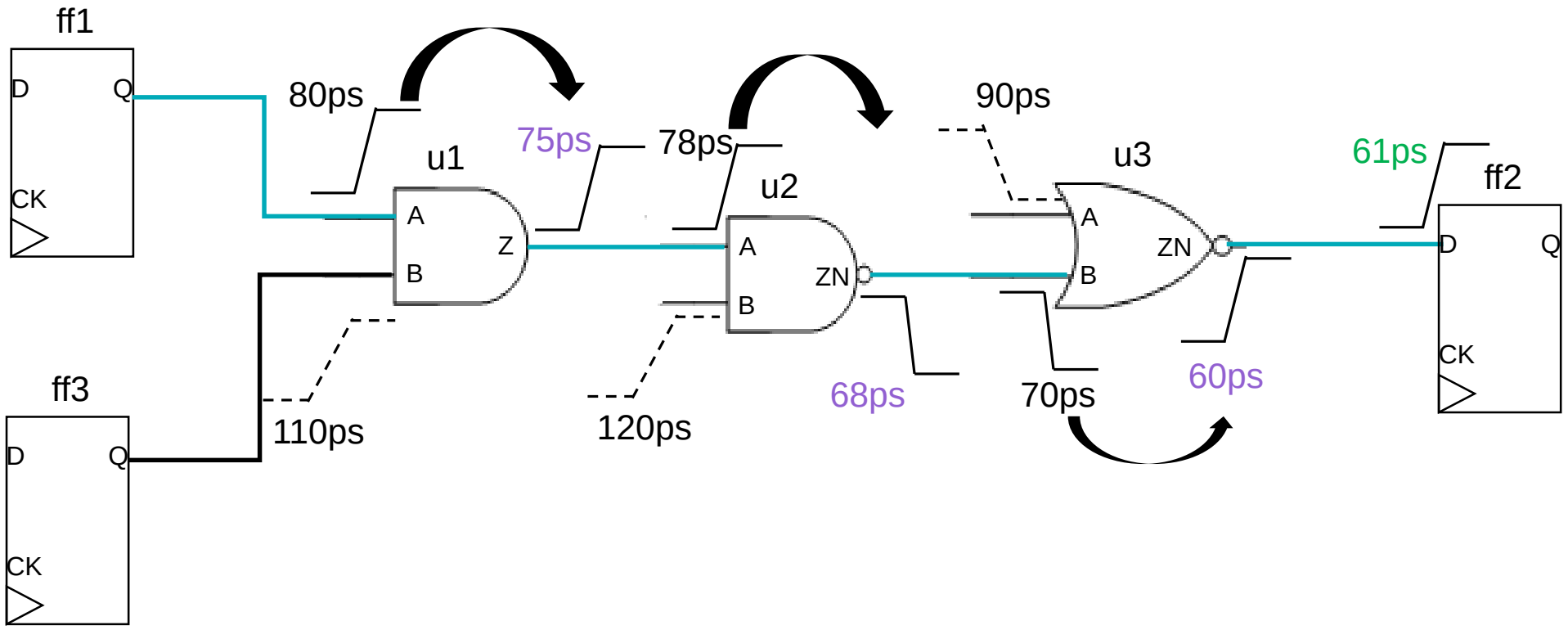
GBA STA



Path: ff1/CK → ff1/Q → u1/A → u1/Z → u2/A → u2/ZN → u3/B → u3/ZN → ff2/D

Merging of worst transitions at merge nodes ➔ pessimism

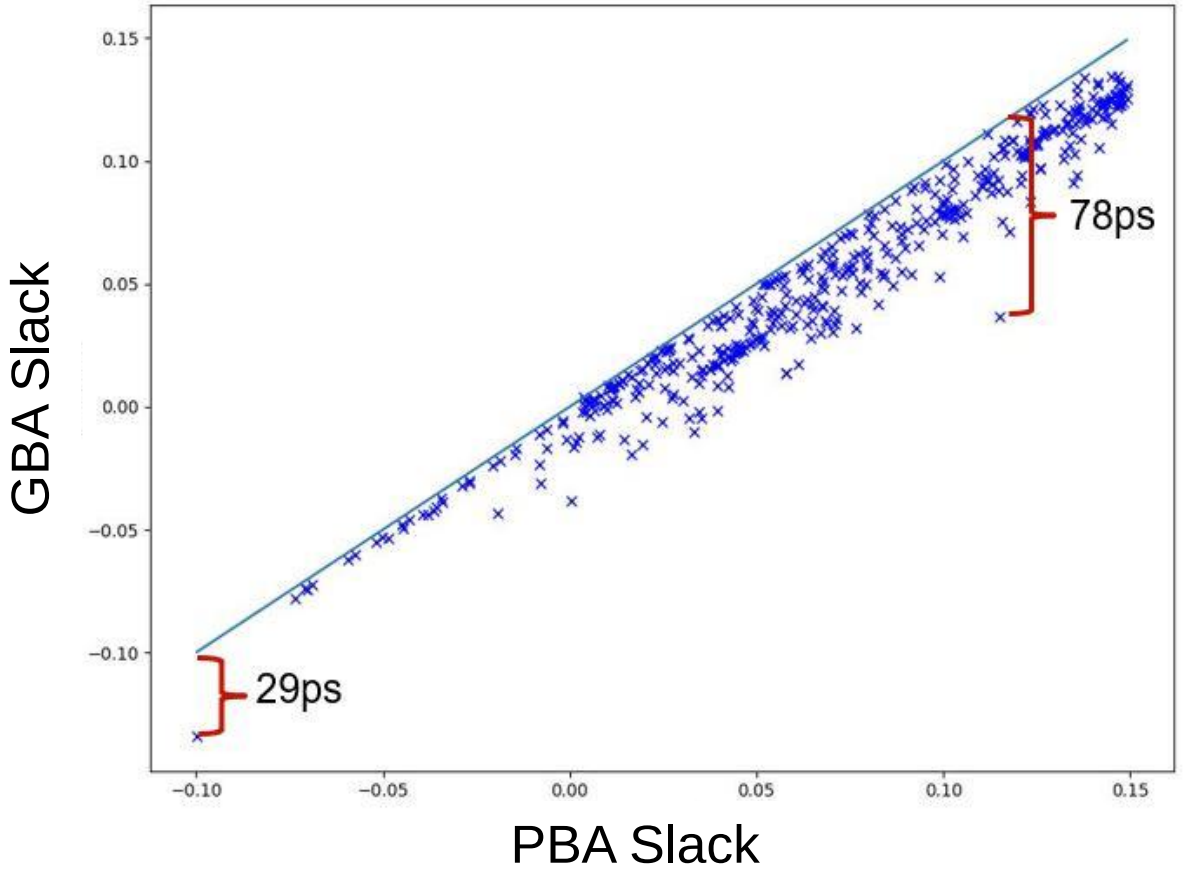
PBA STA



Path: ff1/CK → ff1/Q → u1/A → u1/Z → u2/A → u2/ZN → u3/B → u3/ZN → ff2/D
Actual transition propagation through nodes

PBA in Digital Implementation: Opportunities

- Achieve signoff-quality timing and power for best-on-class power performance and area (PPA)
- Large gap between GBA and PBA at advanced nodes
  - 78ps delta is ~4-5 stages of logic at 7nm
- Rich optimization moves as compared to ECO signoff
  - Advance logic restructuring
  - Layer promotion/NDR with re-route and rebuffering
  - Flexible clock and data optimization
  - Legalization- and eco-routing-aware
- Prevents over-design (area / power loss)



PBA in Digital Implementation: Challenges

- “True” PBA analysis requires exponential path tracing
- Massive increase in memory and runtime
  - Overhead can be ~50% of GBA runtime
- “True” PBA exhaustive makes TAT infeasible on large-scale designs with 1-10M instances
  - PBA path is an alternative (retimes the worst GBA path using PBA analysis)
  - Practical considerations limit design companies to analyze top-K paths (e.g., K = 10000)
- Incremental STA using PBA in implementation flows is tricky
  - Fine balance of runtime vs. correct analyses

Previous Works

- PBA runtime improvement heuristics
  - Bai et al. propose PBA path, top-K path analyses
  - Gupta et al. propose IPBA, an infinite-depth parallel traversal of timing graph using BFS and speedup stage-wise computation
  - Shyamsukha et al. perform selective PBA analyses based on statistical and non-statistical path characteristics
  - Wrixon et al. propose a thresholding mechanism while merging timing values

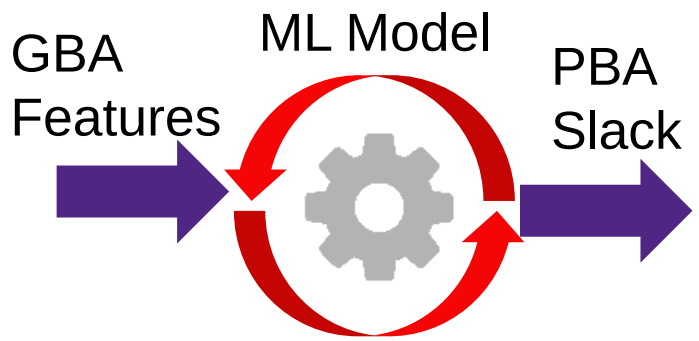
- ML-based heuristics
  - Huang and Wong create a task graph and use MapReduce to massive scale PBA analyses
  - Kahng et al. predict PBA timing using GBA features using a bigram approach to capture path timing features
    - Closest to our work
    - Large runtime for deployment in implementation flows

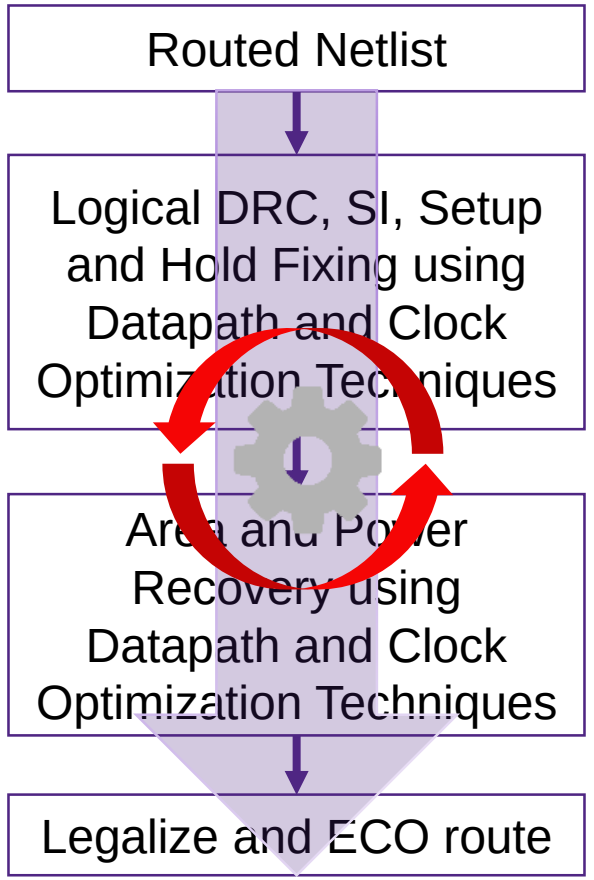
Our Key Contributions

- Our is the ***first*** application of ML paradigm to *optimize* post-routed designs using predicted PBA slack using a *ML-augmented* GBA flow
- We propose *on-the-fly training* methodology and *insights* on choice of features and model algorithm for our prediction task
- We apply *engineering strategies* on model guidance for delay, area and power recovery to reduce GBA flow over-design
- We *integrate* our methodology in a *commercial EDA tool* and present non-trivial PPA gains on *real industrial designs* across 5nm – 16nm
- Our experimental results have a moderate runtime overhead ~3% (vs. GBA) for up to 11.7% improvement in leakage power and 1.16% in total power
  - Baseline GBA runs state-of-the-art power flows on industrial designs where every 1% total power matters

Problem Formulation

- Design ML-based predictor of PBA slack given GBA features
- Predict MCMM PBA slack across all critical paths
- Separate models for delay and recovery
- Integrate models in a commercial post-route optimization flow
- Adjust GBA timing on critical paths based on ML model predictions





On-The-Fly Training

- Key innovation in our work

- Eliminate need for two-pass training and inference flow requirements for supervised modeling
  - Causes deployment challenges at customers due to data management

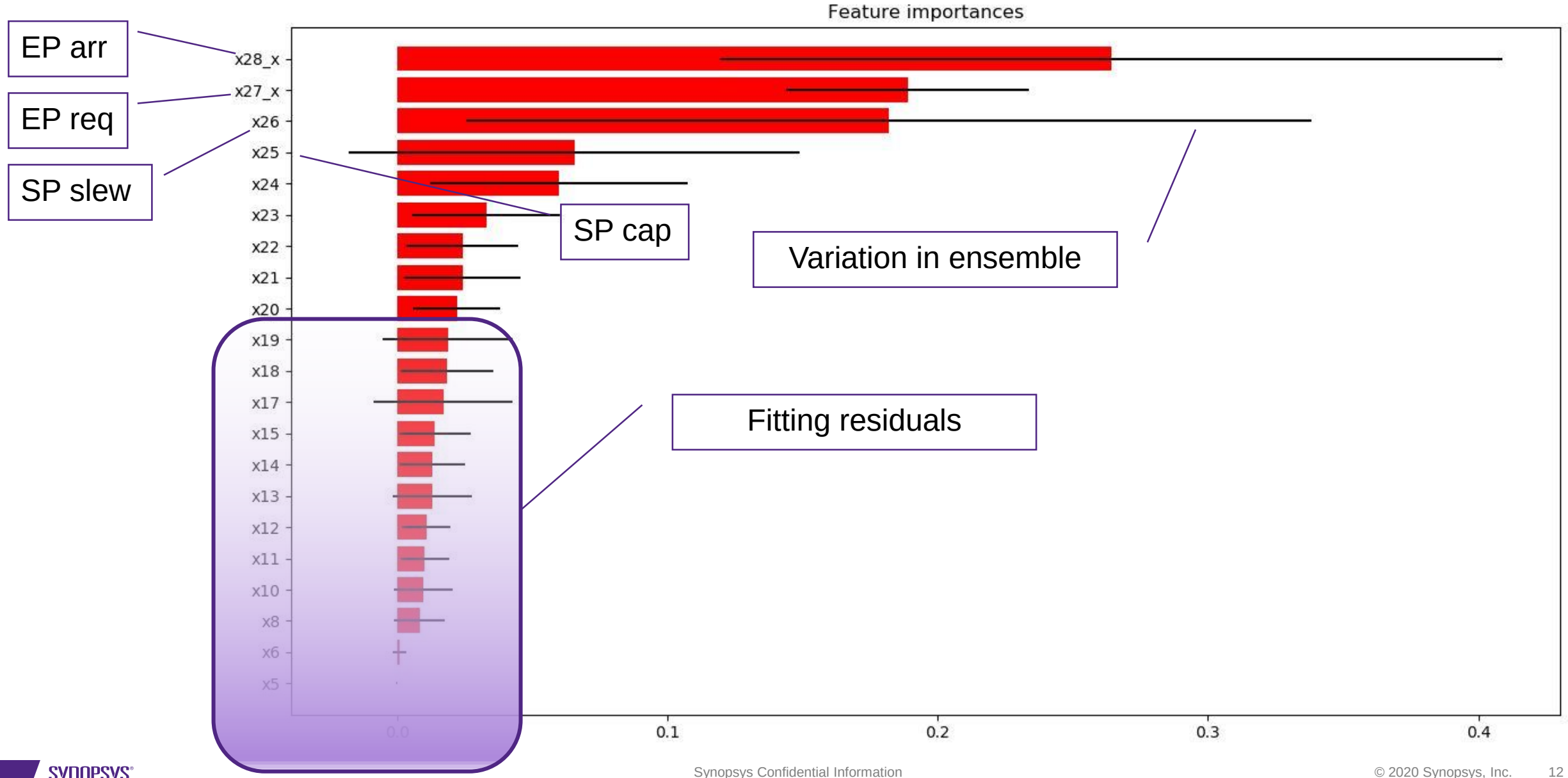
- Develop a one-pass flow
  - Model trained when large changes to netlist occurs that shifts PBA labels
  - Model inferred / refreshed at strategic points in the optimization flow

- Requires a light-weight and accurate training methodology
  - Low overhead of feature extraction
  - Fast and parallel algorithms for training
  - Runtime overhead of above <5% of a baseline GBA flow
  - Cost function that accurately models design PPA

Our Features

Category	#	Features Examples	Justification
Physical Context	4	SP(x, y); EP(x, y); WL(X); WL (Y)	Discriminates path layouts
Logical Context	4	#stages in critical path; max fanout; avg fanout; avg drive resistance	Indicates extent of GBA-PBA divergence
Timing Context	9	SP cap, slew; EP cap, slew; cell and net delays; ratio of net to total delay; avg slew, cap	Discriminates between path timing characteristics
Physical Constraints	2	#dont_touch,; #size_only	Indicates extent of GBA-PBA divergence
Timing Constraints	4	#maxtran violators on critical paths; EP arrival, required; MCMM scenario	Discriminates types of paths and indicates the extent of GBA-PBA divergence

Feature Importances



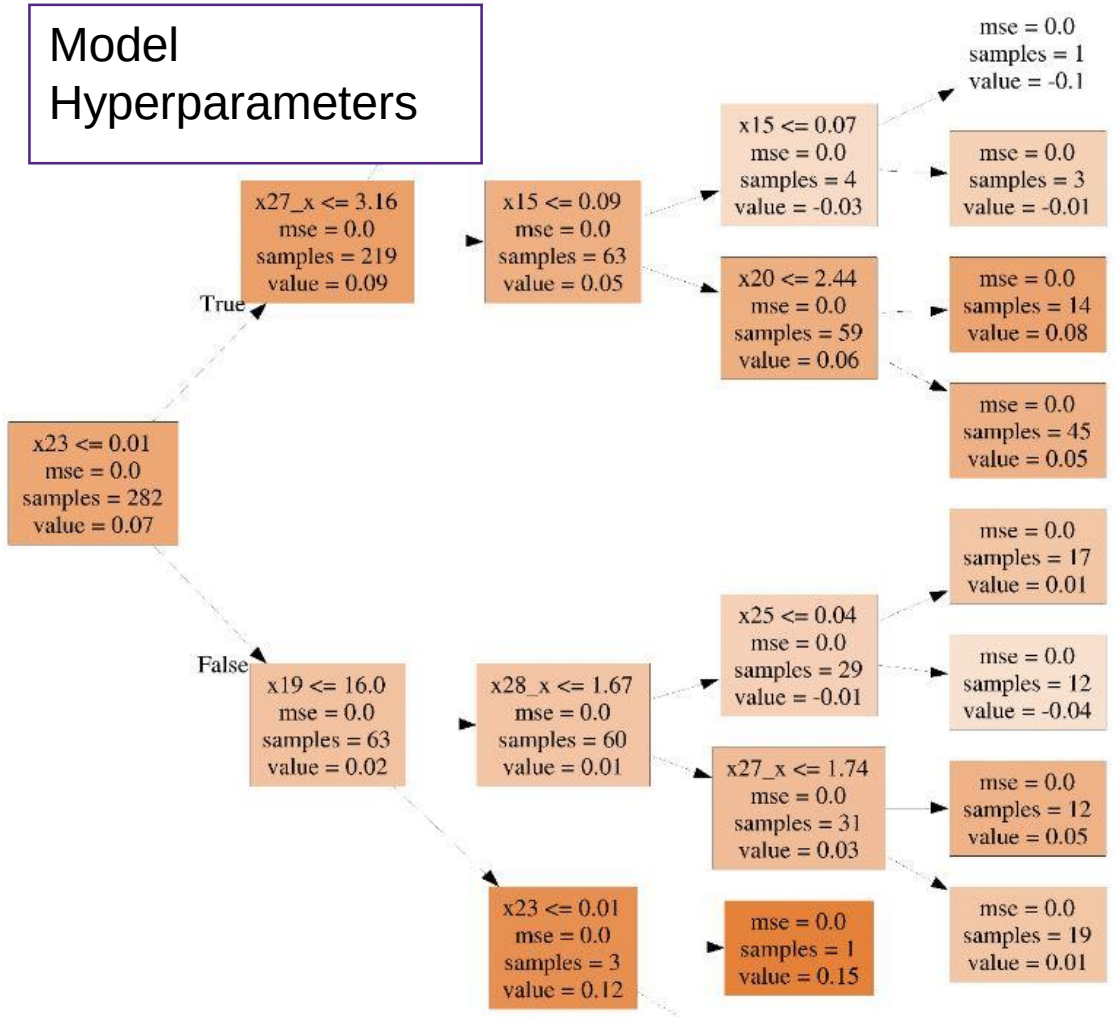
Machine Learning Algorithm

$$\hat{y}^{PBA} = f(\mathbf{X}^{\mathbf{GBA}}, \theta)$$

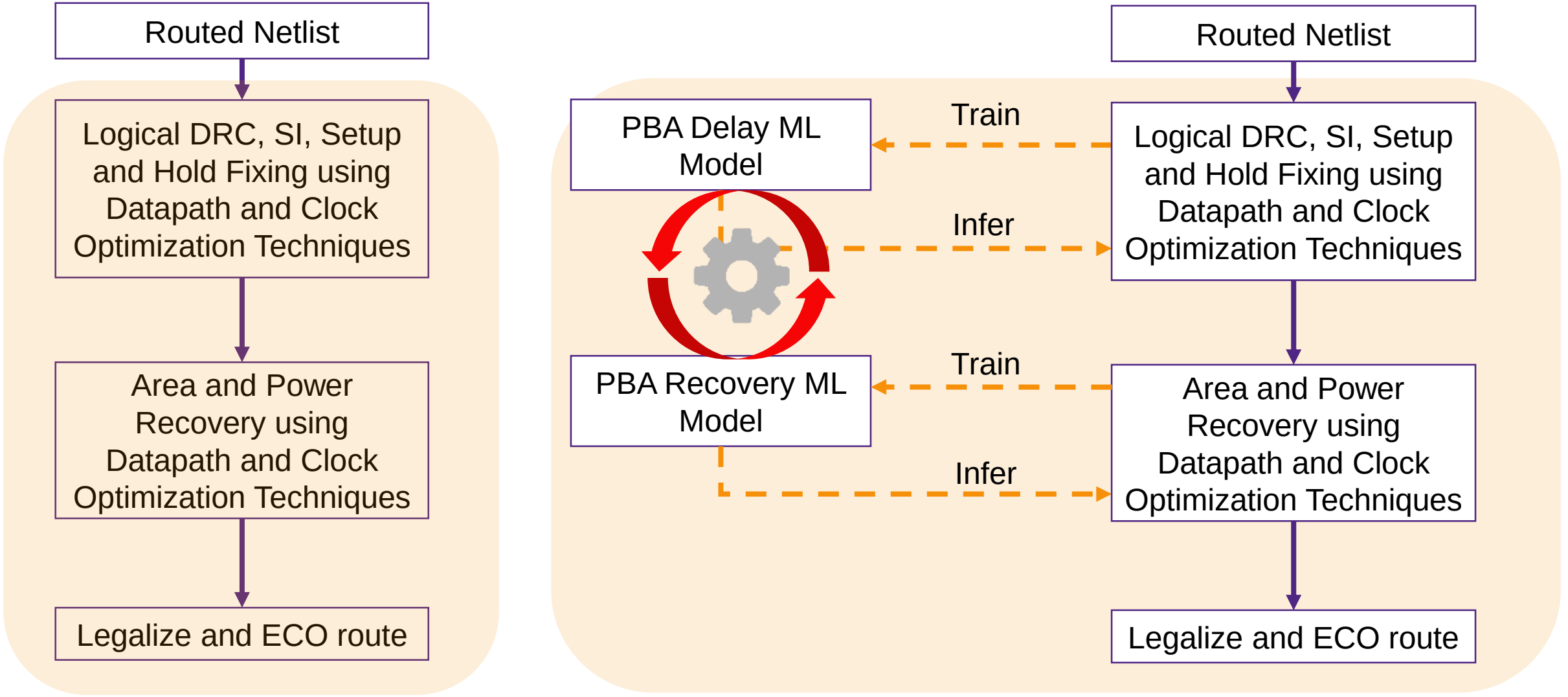
GBA features

Model Hyperparameters

- Gradient-boosted Random Forest regressor
  - Fast to train and accurate
  - Easier to debug outliers as compared to DNNs or othe complex models
  - Intuitive hyperparameters and easier to tune as compared to those of DNNs
- Algorithm bootstrapped using 50-70% of training dataset
- At each node, randomly sample 50-70% of features
- Select feature with the largest information gain
- Prune trees that violate max depth or min #child nodes criteria
- Objective is to minimize RMSE between GBA and PBA slack → accurate indicator of design TNS



ML-Guided Post-Route Optimization



Baseline GBA Flow

ML-Augmented GBA Flow

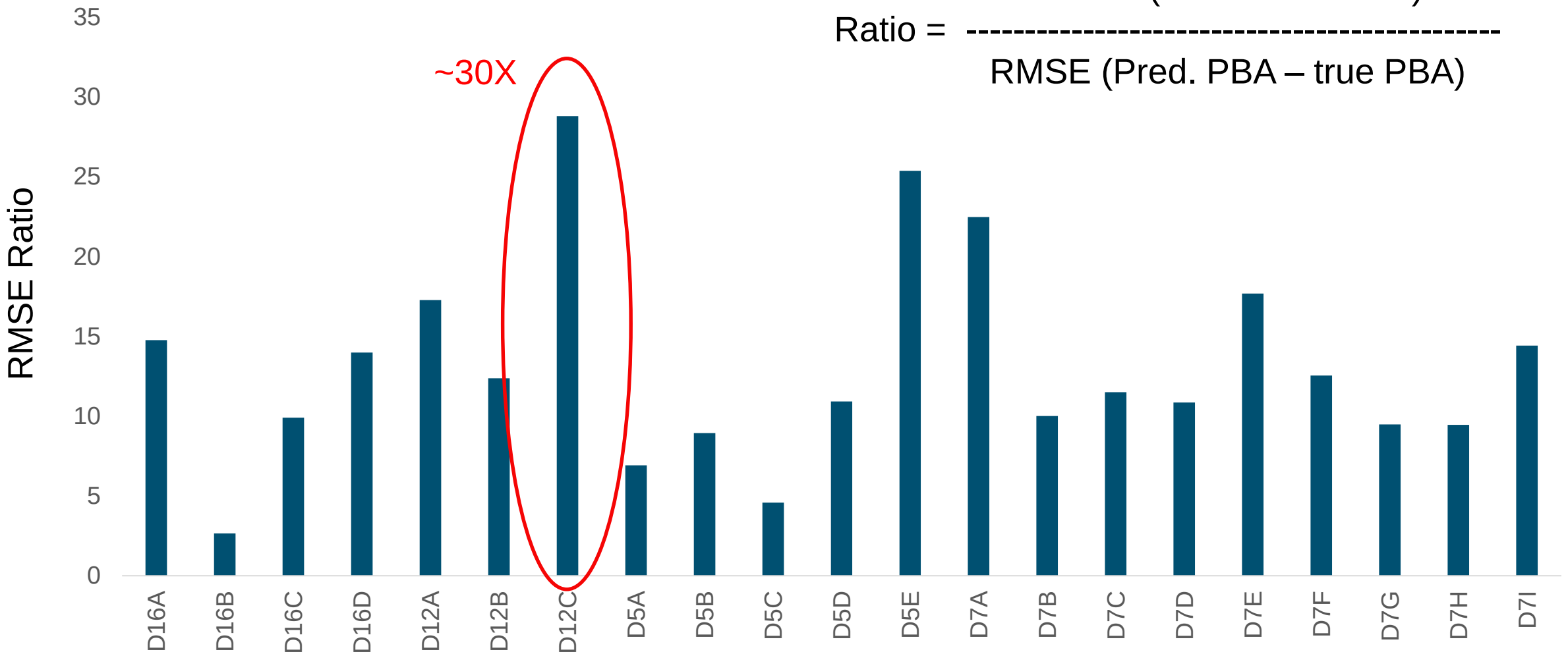
Experimental Setup

- Modeling implemented in C++ and Python; 80%-20% train-test split
- Model trained using PBA path for fast turnaround of experiments
- Integrate in post-route optimization flow of a commercial EDA tool
- Experiments run on 21 industrial designs and run on Intel Xeon 16-core 2.6GHz machines

Node	Designs	#	#Instances Range	Power Type
16nm	D16A – D16D	4	161K – 2.4M	Total power
12nm	D12A – D12C	3	130K – 855K	Total power
7nm	D7A – D7E D7F – D7I	5 4	331K – 1.5M 272K – 492K	Total power Leakage power
5nm	D5A – D5D D5E	4 1	94K – 359K 67K	Total power Leakage power

ML Model Accuracy: Delay

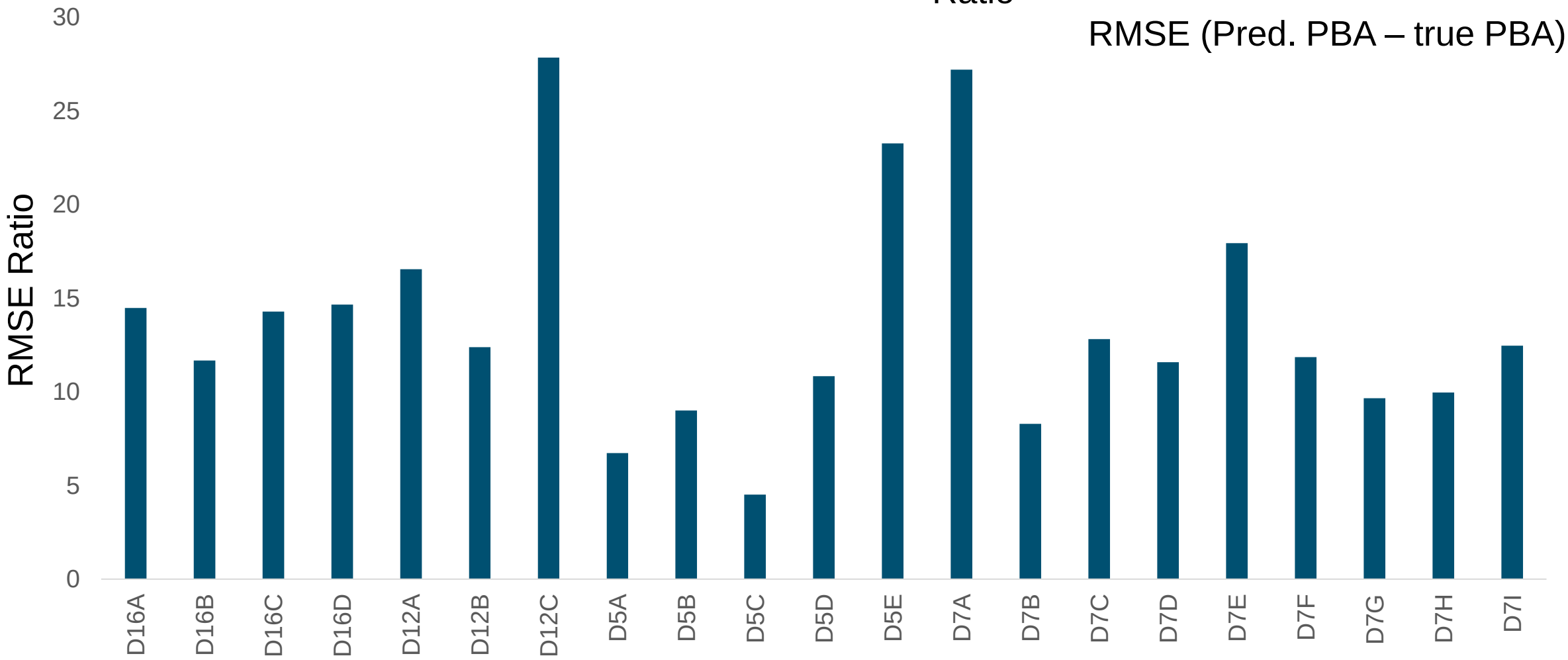
$$\text{Ratio} = \frac{\text{RMSE (GBA – true PBA)}}{\text{RMSE (Pred. PBA – true PBA)}}$$



D16B and D5C have small GBA – PBA gap

ML Model Accuracy: Recovery

$$\text{Ratio} = \frac{\text{RMSE (GBA – true PBA)}}{\text{RMSE (Pred. PBA – true PBA)}}$$



D5C have small GBA – PBA gap

Impact on Post-Route Optimization

- PBA reduces delay pessimism
  - Fewer #instances selected for optimization during delay fixing as compared to GBA
  - Larger #instances selected for optimization during area, power recovery as compared to GBA

Design	Attempted		Accepted		Runtime(s)	
	GBA	ML	GBA	ML	GBA	ML
Delay						
D16D	53K	47K	27K	16K	2075	1770
D12A	294K	247K	181K	130K	11119	10541
D7B	180K	168K	8.1K	6.3K	631	437
D7F	58K	49K	42K	25K	1439	892
D5C	12K	8.3K	6.1K	2.6K	163	143
Recovery						
D16C	1.75M	1.83M	299K	358K	3025	3745
D16D	1.65M	1.66M	194K	239K	3467	3626
D12A	49	1.93M	10	357K	~0	8065
D7B	3.74M	3.74M	364K	415K	3658	4091
D7F	694K	793K	206K	260K	945	1823
D5C	125K	129K	40K	41K	291	375

Less runtime in delay optimization

More runtime in area, power optimization

Small GBA-PBA gap ➔ insignificant impact on D5C

Post-Route Optimization PPA

Design	GBA			ML		
	TNS (ns)	Area (μm^2)	Power (mW)	TNS (ns)	Area (μm^2)	Power (mW)
D16C	-14.3	813K	3567	-19.6 (0.23%)	812K (-0.02%)	3526 (-1.16%)
D16D	-2.64	701K	762	-13.1 (0.16%)	700K (-0.04%)	3626 (-0.87%)
D12A$^{\Upsilon}$	-99.4	286K	30.4	-81.3 (-0.41%)	283K (-0.92%)	28.2 (-7.19%)
D7B	-4.16	291K	1145	-6.83 (0.20%)	291K (-0.07%)	1142 (-0.23%)
D7F$^{\Upsilon}$	-0.25	74K	9.97	-0.55 (0.13%)	72K (-0.86%)	8.80 (-11.7%)
D5C	-1.22	11K	201	-1.14 (-0.08%)	10K (-0.39%)	202 (0.25%)

Υ → Leakage-only design

- TNS is normalized to path delay for unbiased comparisons
- ML-augmented GBA flow improves area and power as expected
- Sources of TNS degradation
  - Inaccuracies due to model reuse
  - Model limitation as PBA path is used for training

Statistics of Overall PPA, Runtime

Statistics	Area	Leakage	Dynamic	DRC
Mean (μ)	-0.12%	-2.30%	-0.14%	-4.76%
Avg μ/σ	-2.94	-3.17	-2.95	-0.37
%Win	81%	95%	79%	10%
%Neutral	0%	0%	21%	80%

Statistics	WNS	TNS	THV	#MaxTran Violations	Runtime
Mean (μ)	+0.02%	+0.07%	+6.27%	-3.12%	+3.37%
Avg μ/σ	+0.11	+1.88	+0.74	-0.88	+1.78
%Win	48%	81%	43%	76%	43%
%Neutral	38%	0%	10%	10%	19%

Summary and Future Work

- PPA optimization at advanced process nodes is challenging in traditional GBA-based implementation flows
- We demonstrate a fast and accurate ML-enabled methodology to learn PBA timing and apply in a ML-augmented post-route optimization flow
- Our detailed insights on feature engineering, model selection hopefully helps other ML projects in digital implementation
- We achieve up to 11.7% leakage and 1.16% total power reduction across multiple industrial designs from top semiconductor companies spanning from 16nm down to 5nm
- Our ongoing works
  - Improve timing outliers
  - Develop ML-augmented pre-route GBA flows

SYNOPSYS®
Silicon to Software™

Thank You