

Machine Learning for Digital Design Automation

Ioannis Savidis
Pratik Shrestha



1

Drexel University – Electrical & Computer Engineering

2

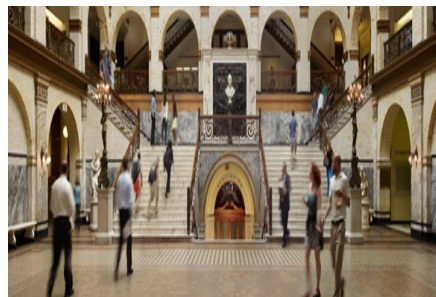


Drexel University:

- **Founded** in 1891 by financier and philanthropist Anthony J. Drexel
- **Location:** four campuses: 3 in Philadelphia, 1 in New Jersey (Mt. Laurel)
- **Student Enrollment:** 15,346 undergraduates, 8,859 graduate and professional students
- **Student Geographic Distribution:** Students come from 50 U.S. states and 130 foreign countries. Nearly 8% are international students



Bossone building
(home to College of Engineering labs)



2

DREXEL UNIVERSITY - IOANNIS SAVIDIS GROUP

3



Degrees: B.S.E., Duke University
M.S., University of Rochester
Ph.D., University of Rochester (2013)

Research Interests

Analysis, modeling, and design methodologies for high performance digital and mixed-signal integrated circuits; Emerging integrated circuit technologies; Electrical and thermal modeling and characterization, signal and power integrity, and power and clock delivery for 3-D IC technologies; hardware security (obfuscation and side-channel analysis); algorithms and methodologies for design automation including ML/AI based optimization; On-chip power management; Low-power circuit techniques; Algorithms and methodologies for secure IC design

LABORATORY & TEAM

- **Eight Ph.D. students**
 - Alec Aversa – Sequential digital circuit obfuscation
 - Saran Phatharodom – Digital obfuscation metrics
 - Jeff Wu – Application of ML/AI to analog/RF IC design
 - Ziyi Chen – Analog IP protection
 - Ashish Sharma – Heterogeneous circuit integration
 - Pratik Shrestha – Digital security and application of ML/AI to digital IC design
 - Nnaemeka Achebe – Application of ML to RF design
 - Amit Varde – EDA foundational modeling, ML based analog design
- **One M.S. student**
 - David Binder – dataset generation for digital IC ML/AI EDA problems
- **2,000 square feet of dedicated research space**
- **Access to leading CAD software packages:** Cadence (Virtuoso, Innovus, Assura, Quantus, Genus, etc.), Synopsys (FusionCompiler, PrimeTime, Hspice, Taurus, etc.), and Siemens Mentor Graphics (Calibre)



3

ICE Research:

Heterogeneous & Reconfigurable Integration for Edge Compute



Heterogeneous Chiplet and 3-D Integration

- 3-D/VLSI design methodologies for power and clock network design
- Multi-plane power noise modeling
 - Methodologies to mitigate cross-plane coupling
- Power management for multi-domain, multi-plane delivery
- Clock tree synthesis for heterogeneous device planes
- Multi-physics modeling of electro-thermal-mechanical characteristics of 3-D ICs
- Test vehicle design and characterization (three fabricated and tested ICs)

Power Management

- Near-threshold circuits (NTC) for low-power applications
 - Implement circuit families including CMOS and current mode logic in NTC
- Leakage reuse for multi-voltage domain systems
- Energy efficient heterogeneous DNN accelerators
- Power management for multi-domain delivery

ML/AI Circuit Design

- Applied algorithms for clock tree synthesis
- ML/AI algorithms for analog transistor sizing
 - Classification with adaptive labeling
- SMT based optimization of transistor sizing
- Graph based representation of circuit netlist
- EDA-Learn: framework for circuit data generation and ML modeling

Hardware Security & Trust

- Real time Trojan detection
- FPGA security and IP protection
 - Detection of hardware Trojans
 - Real time side-channel monitoring
- Attack prevention with trusted design
 - Metrics to quantify security
 - Algorithms and methodologies to obfuscate digital circuits
- Protection of analog circuits
 - Circuit redaction using field programmable analog arrays
 - Key-based parameter obfuscation
- Heterogeneous IC security (chiplet/3D)
 - Trojan detection
 - Side-channel analysis
 - Secure multi-plane communication

Objective:
**Secure Heterogeneously
Integrated Circuits and Systems
for Edge Compute**



4

Outline of Presentation

6

- Introduction to Electronic Design Automation
- Machine learning techniques
- Case studies
- Standardizing ML for digital EDA
- Conclusions



6

Outline of Presentation

7

- Introduction to Electronic Design Automation
- Machine learning techniques
- Case studies
- Standardizing ML for digital EDA
- Conclusions

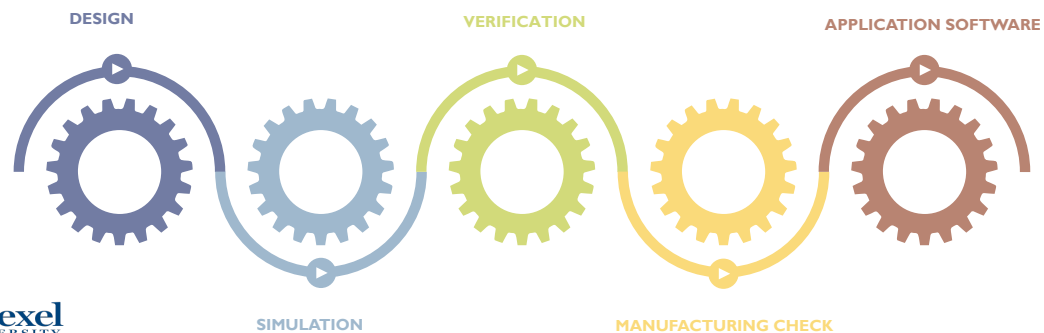


7

Background: Electronic Design Automation (EDA)

8

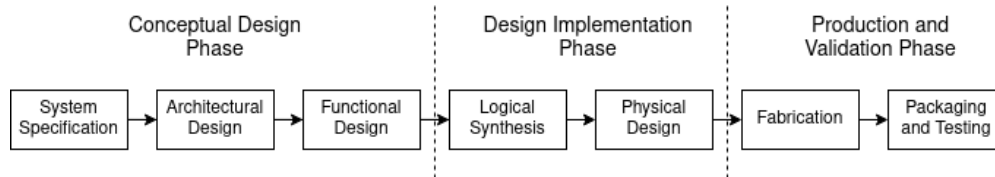
- EDA: a system of software solutions for the design of integrated circuits
- A wide range of applications:
 - High-performance Computing
 - Autonomous vehicle
 - IoT
 - AI
 - ...
- Primary Tools/Applications:



8

IC Design Cycle

10



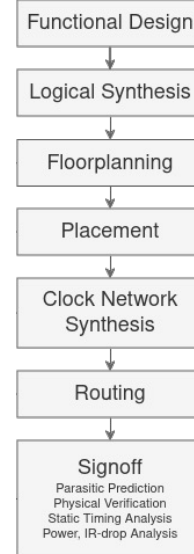
- Divided into three phases
 - **Conceptual design:** requirement definition, high-level structural outlining, and functionality design
 - **Design implementation:** translation of conceptual design into a logical representation and then a physical layout
 - **Production and validation:** physical fabrication, packaging, and testing
- Design implementation phase has the highest scope of automation

10

Digital IC Design Automation Flow

11

- RTL-to-GDSII flow
 - Frontend: technology independent standardized design descriptions
 - Examples: VHDL, Verilog
 - Backend: physical implementation of circuits
 - Fabs provide Process Development Toolkits (PDK) and simulation models for fab processes



11

Digital IC Design Automation Flow – Conceptual Design

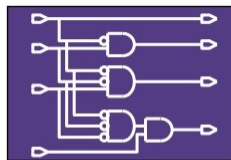
12

- Functional Design using Hardware Description Language
- Logical Synthesis
 - Convert hardware description language (HDL) into technology (PDK) specific standard cells
 - Translation + Mapping + Optimization

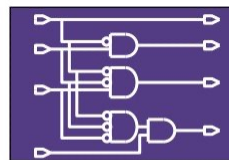
```

residue = 16'h0000;
if (high_bits == 2'b10)
    residue = state_table[index];
else
    state_table[index] =
    16'h0000;
    
```

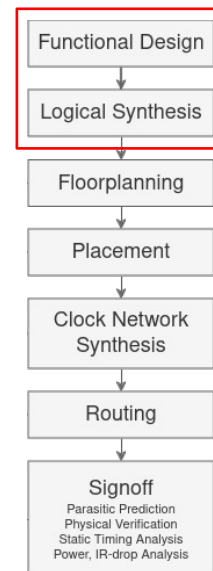
Hardware Description Language (HDL)



Target Technology (standard cells)



Generic Boolean (GTECH)

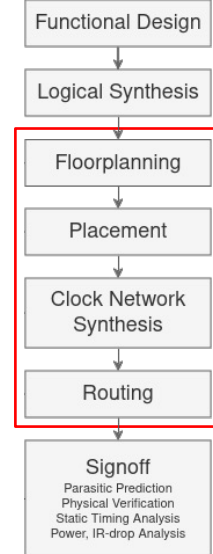
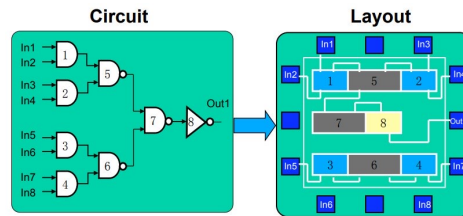


12

Digital IC Design Automation Flow – Design Implementation

13

- Functional Design using Hardware Description Language
- Logical Synthesis
- Physical Design
 - Transform a technology mapped logical circuit into actual layout to be fabricated
 - Optimize placement of cells and interconnections to optimize performance, power, and area constraints

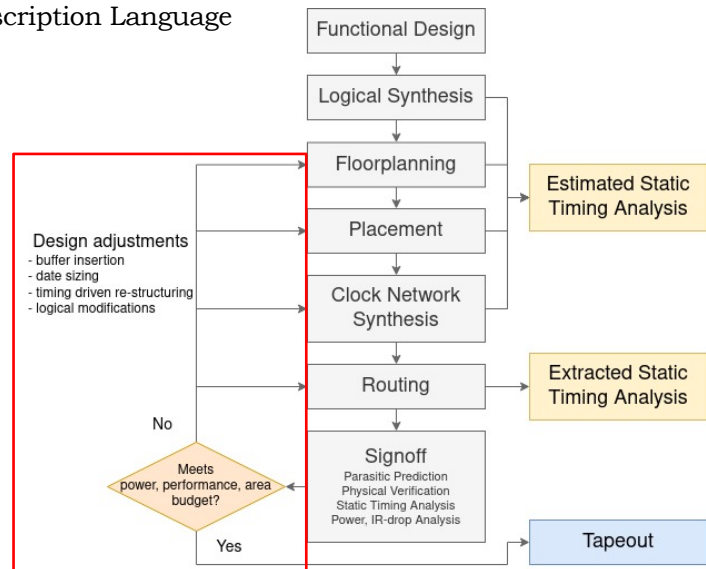


13

Digital IC Design Automation Flow – Design Adjustments

14

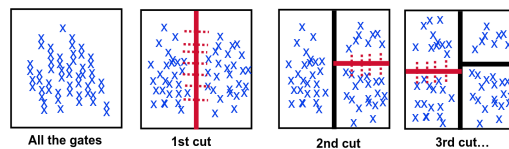
- Functional Design using Hardware Description Language
- Logical Synthesis
- Physical Design
- Verification and signoff
 - Confirm circuit conforms with power, performance, area budget
 - Fix circuit if design requirements not met



14

Overview of Heuristic Approaches for Design Synthesis

- Rule based (heuristics) + Optimization
 - Time and resource expensive
- Example: Recursive Placement using "Min-Cut" Approach
 - **Rule based (heuristics):** cut the circuit into 2 pieces, partition gates across the 2 sides
Continue executing recursively with resulting partitions of each step



- Optimization:** minimize wiring between partitions
 Provided each net has a cost c_{ij} , minimize total sum of costs of nets across cut

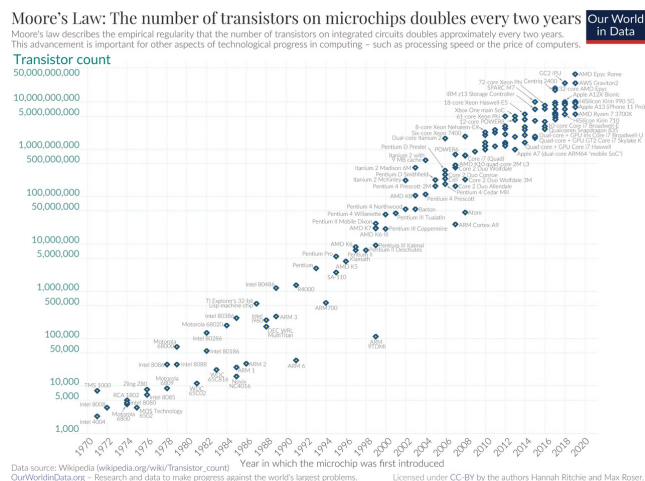
$$\text{minimize } T = \sum_{\text{cut nets}} c_{ij}$$



15

Challenges of Digital IC Synthesis

- Complexity of design tasks



Source: en.wikipedia.org/wiki/Moore's_law



16

Challenges of Digital IC Synthesis

17

- Complexity of design tasks
 - Apple M3: 25 billion transistors
 - Apple M3 Pro: 37 billion transistors
 - Apple M3 Max: 92 billion transistors
- Complexity of computational platform
 - SoCs with mixed-signal functionality



Source: www.apple.com/newsroom/2023/10/apple-unveils-m3-m3-pro-and-m3-max-the-most-advanced-chips-for-a-personal-computer
 Source: [Die walkthrough: Alder Lake-S/P and a touch of Zen 3 \(substack.com\)](https://www.substack.com/p/die-walkthrough-alders-lake-s-p-and-a-touch-of-zen-3)

17

Challenges of Digital IC Synthesis

18

- Complexity of design tasks
- Technology node scaling challenges



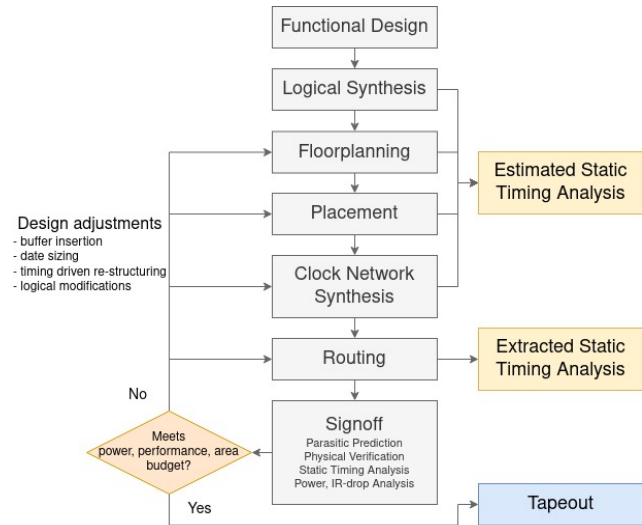
Source: www.digitimes.com/news/a20200702PD200.html

18

Challenges of Digital IC Synthesis

19

- Complexity of design tasks
- Technology node scaling challenges
- Iterative nature
 - Optimal design requires multiple iterations
 - Change in upstream parameters have significant downstream effects
- Expensive design flow
 - One iteration may take from hours to days to complete

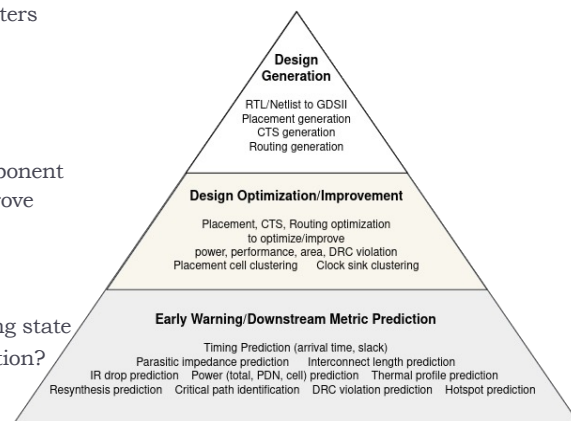


19

Machine Learning for EDA

20

- Prediction
 - Model the design space and predict circuit parameters
 - Can we predict post placement wirelength before placement?
- Optimization
 - Improve on a circuit state or state of a circuit component
 - Can we use previous placement knowledge to improve current placed design?
- Generation
 - Create a circuit component without any pre-existing state
 - Can we generate a completely new placement solution?

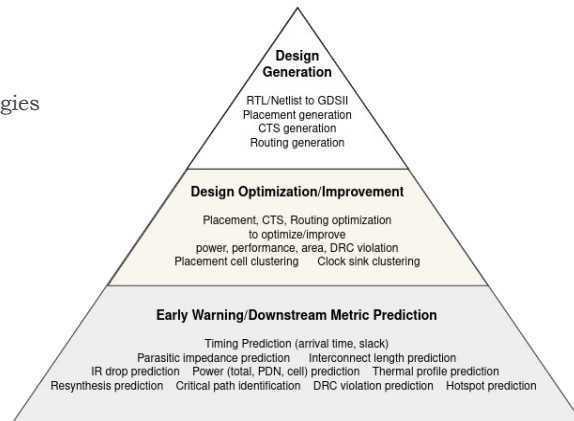


20

Machine Learning for EDA

21

- Transfer Learning
 - Reuse learned knowledge from previous designs and nodes to accelerate model development
 - Can we adapt pretrained models to new technologies and designs with minimal fine-tuning?
- Large Language Models (LLM)
 - Used across the pyramid to assist reasoning, debugging, and flow steering throughout the EDA design process

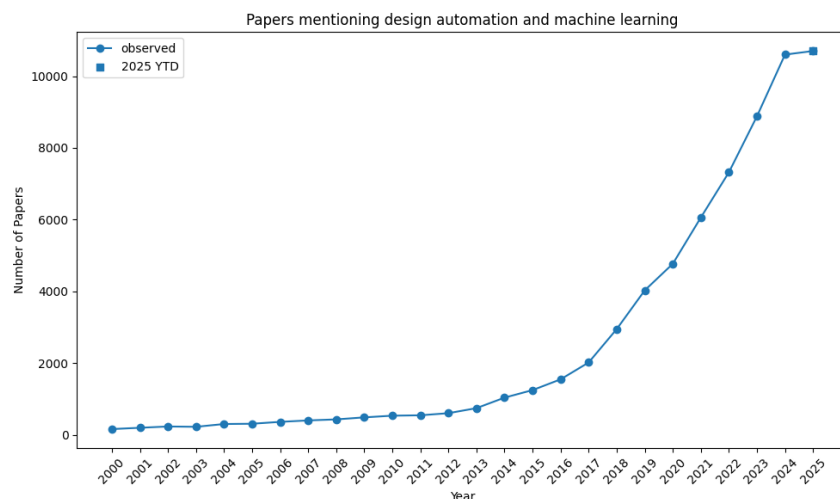


21

Interests in ML for EDA

22

- Papers mentioning design automation and machine learning



22

Outline of Presentation

23

- Introduction to Electronic Design Automation
- Machine learning techniques
- Case studies
- Standardizing ML for digital EDA
- Conclusions

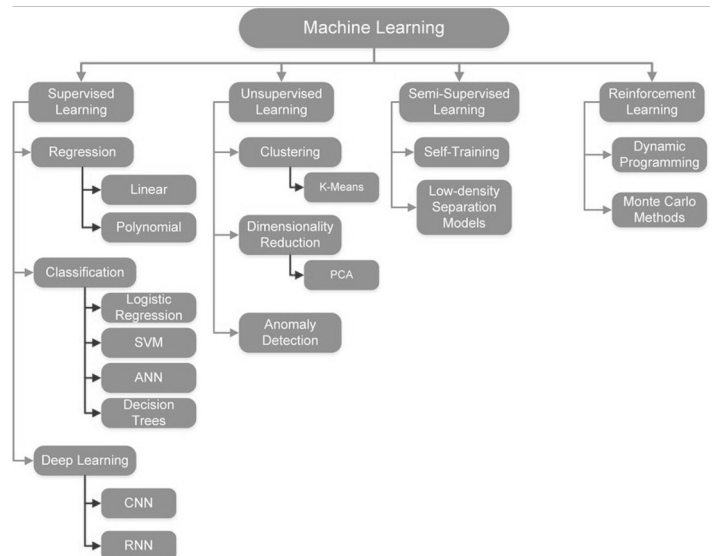


23

Families of Machine Learning Algorithms

24

- Train models to learn from data
- Leverage information from data to improve performance on prediction and generation tasks
- Diverse algorithm choices
 - Based on application
 - Based on complexity
 - Based on data format



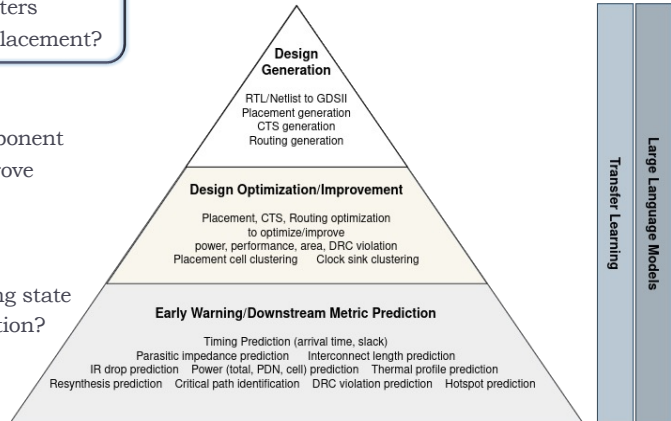
A. Moubayed, M. Injadat, A. B. Nassif, H. Lutfiyya and A. Shami, "E-Learning: Challenges and Research Opportunities Using Machine Learning & Data Analytics," *IEEE Access*, Vol. 6, No. 1, pp. 39117-39138, July 2018

24

Machine Learning for EDA

25

- Prediction
 - Model the design space and predict circuit parameters
 - Can we predict post placement wirelength before placement?
- Optimization
 - Improve on a circuit state or state of a circuit component
 - Can we use previous placement knowledge to improve current placed design?
- Generation
 - Create a circuit component without any pre-existing state
 - Can we generate a completely new placement solution?



25

Regression

26

- Model the relationship between a dependent (target) variable and one or more independent (predictor) variables

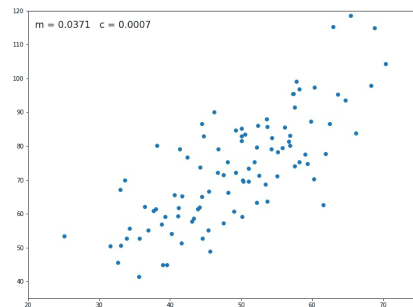
Linear Regression

- A linear regression model is given by

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n + \epsilon$$

where

- y is the dependent variable
- x_i are independent variables
- β_i are coefficients
- ϵ is the error term
- Coefficients are optimized by minimizing error between actual value (y) and predicted value (\hat{y}) over multiple epochs



Source: keytodatascience.com

26

Classification

27

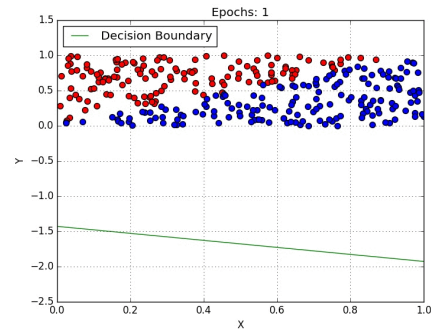
- Categorize new, unseen data based on learning from a feature dataset with known labels

Logistic Regression

- Linear regression + logistic (sigmoid) function
- Linear regression backbone given by

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n + \epsilon$$
- Activation with the Sigmoid Function
 - Sigmoid function maps linear combination (y) to a (0,1) range
 - Interpret Sigmoid function as the probability of the positive class

$$\sigma(y) = \frac{1}{1 + e^{-y}}$$

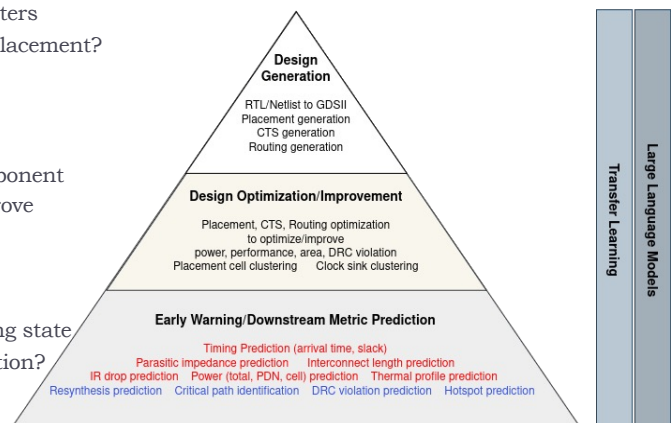


27

Machine Learning for EDA

28

- Prediction
 - Model the design space and predict circuit parameters
 - Can we predict post placement wirelength before placement?
 - **Regression**, **Classification**
- Optimization
 - Improve on a circuit state or state of a circuit component
 - Can we use previous placement knowledge to improve current placed design?
- Generation
 - Create a circuit component without any pre-existing state
 - Can we generate a completely new placement solution?

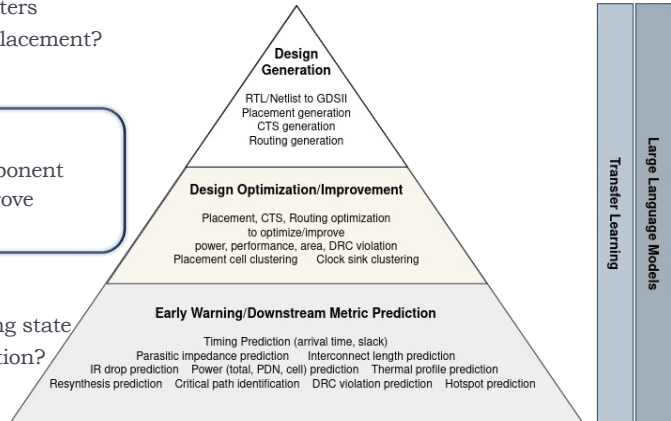


28

Machine Learning for EDA

29

- Prediction
 - Model the design space and predict circuit parameters
 - Can we predict post placement wirelength before placement?
 - **Regression, Classification**
- Optimization
 - Improve on a circuit state or state of a circuit component
 - Can we use previous placement knowledge to improve current placed design?
- Generation
 - Create a circuit component without any pre-existing state
 - Can we generate a completely new placement solution?



29

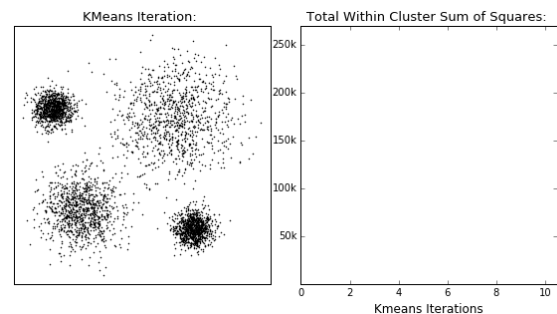
Clustering

30

- Group sets of objects that are similar to each other in a cluster

K-Nearest Neighbor Algorithm

- Classify based on distance between new data point and k nearest known data points
 - Distance metrics:
 - Euclidean distance, Manhattan distance, Hamming distance ...
- Requires storage of all data
 - 'Lazy learning'
- Advantages:
 - Easy to implement and adapt
 - Few hyperparameters
- Limitations:
 - Does not scale well to large dataset with high dimensionality
 - Bottleneck in memory



Source: dashee87.github.io

30

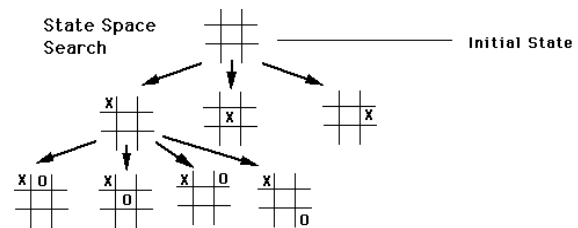
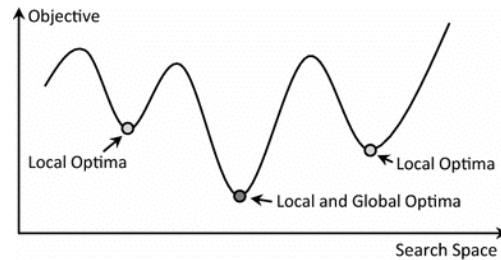
Optimization

31

- Aim to find the most efficient solution from a set of available options

Key Components

- Objective Function
 - Goal of optimization
- Constraints
 - Conditions that the solution must satisfy
- Global optima and local optima
- Searchable state space



31

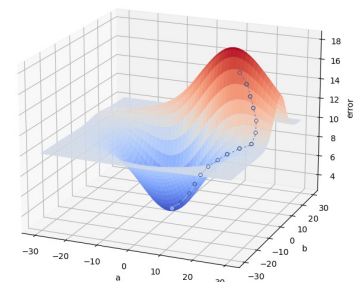
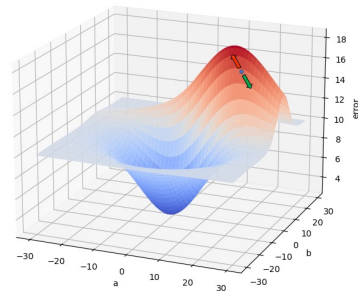
Gradient-based Optimization Algorithms

32

- At each step, search direction is defined by the gradient of the objective function
 - Intuitively, search along direction that reduces cost function at fastest rate
- Gradient descent
 - First-order search of local optimum

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

- Advantages:
 - Theoretical guarantee of optimality
 - Fast execution for each iteration of search
- Limitations:
 - Requires explicit differentiable functions
 - Slow convergence and local minima for non-convex search space



Source: Interactivechaos.com

32

Optimization Algorithms

33

- Gradient-based algorithms
- Heuristic/Classic algorithms
 - Greedy algorithms
 - Divide and conquer
 - Dynamic programming
 - Network flow algorithms
 - Linear/integer programming
 - Evolution-based algorithms
 - Simulated annealing
 - SAT
- Learning-based algorithms
 - Reinforcement learning
 - Surrogate-assisted optimization algorithms

General flow of optimization process

Require: Objective function f
 $x^{(0)} \leftarrow$ random point in the domain of f
for $i = 1, 2, \dots$ **do**
 $\Delta x \leftarrow \pi(f, \{x^{(0)}, \dots, x^{(i-1)}\})$
 if stopping condition is met **then**
 return $x^{(i-1)}$
 end if
 $x^{(i)} \leftarrow x^{(i-1)} + \Delta x$
end for



K. Li, J. Malik, "Learning to optimize," *Proceedings of the International Conference on Learning Representations (ICLR)*, pp. 1-13, April 2017

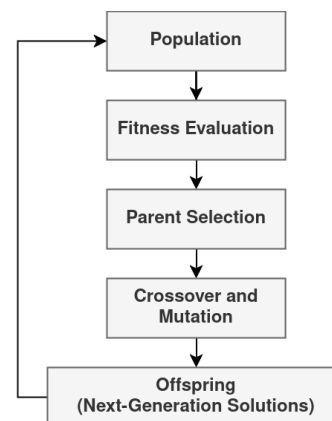
33

Evolutionary Algorithms

34

- Characteristics of evolutionary algorithms
 - Population-Based
 - Fitness-Oriented
 - A fitness parameter to represent the quality of a solution
 - Variation-Driven
 - Crossover and mutation generates variants randomly
- Broad categories
 - Genetic algorithm
 - Particle swarm
 - Differential evolution
 - ...

General flow of evolutionary algorithms



34

Simulated Annealing

35

- A global optimization technique by approximation
- Preferred when search space is discrete
- Parameter: a temperature value that keeps decreasing
 - $\text{temperature} = (\text{initial temperature}) / (\text{iteration} + 1)$
- Steps:
 - Randomly initialize design variables and evaluate function value f_{old}
 - Sample again in the neighboring region and evaluate function value f_{new}
 - Action:
 - If function value improves, accept.
 - If function value worsens, accept with probability $e^{-(f_{\text{new}} - f_{\text{old}}) / \text{temperature}}$
- Advantage:
 - Reduces chance of getting stuck at local optimum since worse candidates are accepted with lower probability.

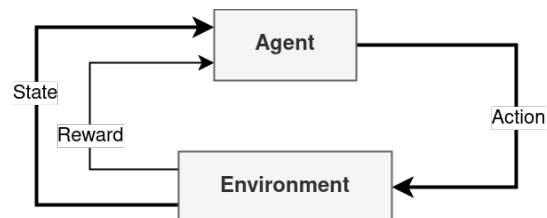


35

Re-enforcement Learning

36

- Agent is trained to identify optimal strategies for decision-making in complex, uncertain environments to achieve its objectives.
- Key Components
 - Agent: AI designer
 - Environment: The world through which the agent operates
 - Action: All the possible actions the agent can execute
 - State: The current conditions as reported by the environment
 - Reward: An immediate feedback from the environment to evaluate the agent's last action



36

Classical Machine Learning Applications

37

- Regression
 - Linear Regression
 - Lasso Regression (L1 Regularization)
 - Ridge Regression (L2 Regularization)
 - Bayesian Linear Regression
- Classification
 - Logistic Regression
 - Decision Trees
 - Random Forest
 - Support Vector Machines (SVM)
 - Naive Bayes
- Clustering
 - ◻ K-Means Clustering
 - ◻ Mean Shift Clustering
 - ◻ Spectral Clustering
- Optimization
 - Gradient Descent
 - Linear/integer programming
 - Simulated Annealing
 - Particle Swarm Optimization
 - Ant Colony Optimization
 - Evolution-based algorithms
 - Genetic Algorithm
 - Re-enforcement Learning
 - Value Iteration
 - Policy Iteration
 - Q-Learning

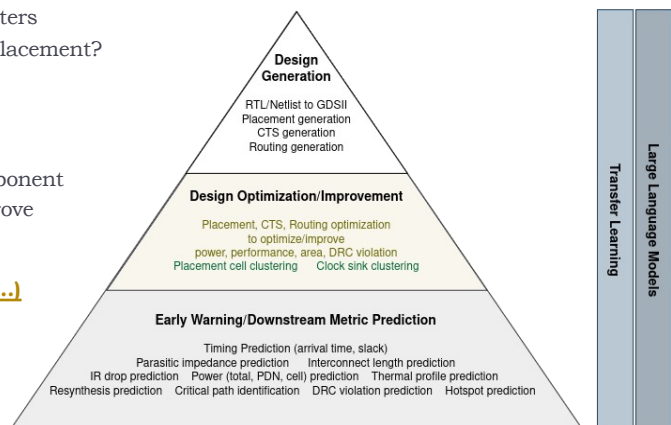


37

Machine Learning for EDA

38

- Prediction
 - Model the design space and predict circuit parameters
 - Can we predict post placement wirelength before placement?
 - **Regression, Classification**
- Optimization
 - Improve on a circuit state or state of a circuit component
 - Can we use previous placement knowledge to improve current placed design?
 - **Clustering, Optimization (Gradient Descent, Simulate Annealing, Re-enforcement Learning ...)**
- Generation
 - Create a circuit component without any pre-existing state
 - Can we generate a completely new placement solution?



38

Classical Machine Learning Applications

39

- Regression
 - Linear Regression
 - Lasso Regression (L1 Regularization)
 - Ridge Regression (L2 Regularization)
 - Bayesian Linear Regression
- Classification
 - Logistic Regression
 - Decision Trees
 - Random Forest
 - Support Vector Machines (SVM)
 - Naive Bayes
- Clustering
 - K-Means Clustering
 - Mean Shift Clustering
 - Spectral Clustering
- Optimization
 - Gradient Descent
 - Linear/integer programming
 - Simulated Annealing
 - Particle Swarm Optimization
 - Ant Colony Optimization
 - Evolution-based algorithms
 - Genetic Algorithm
 - Re-enforcement Learning
 - Value Iteration
 - Policy Iteration
 - Q-Learning

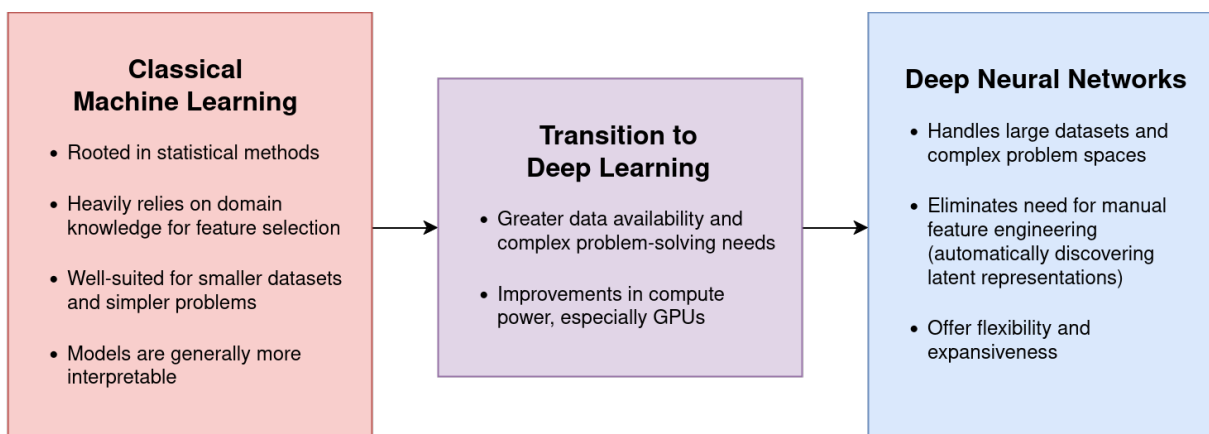
Where are Neural Networks???



39

Machine Learning Progression: Classical ML vs Deep Neural Networks

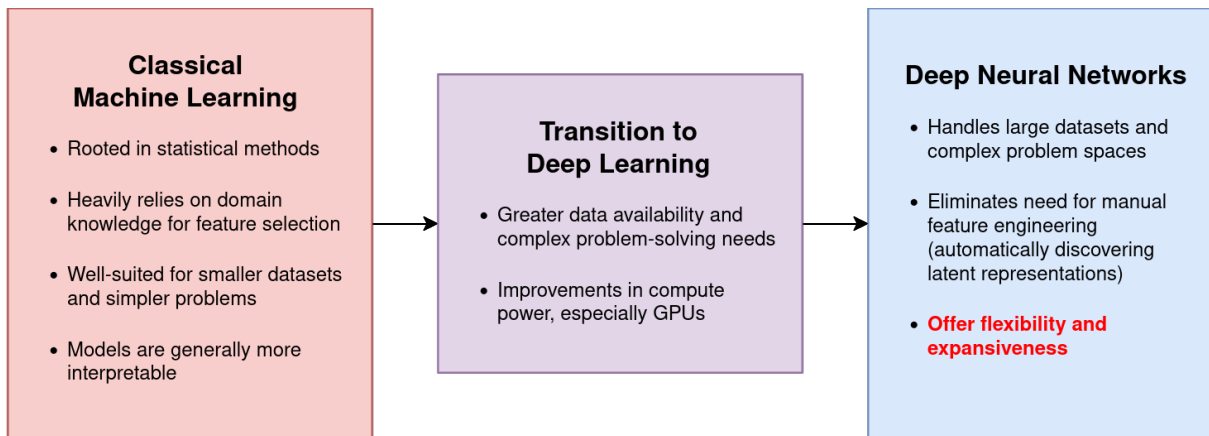
40



40

Machine Learning Progression: Classical ML vs Deep Neural Networks

41



41

Perceptron/Neuron vs Logistic Regression: They are the same!

42

- Deep neural network <-> a stacked network of neurons

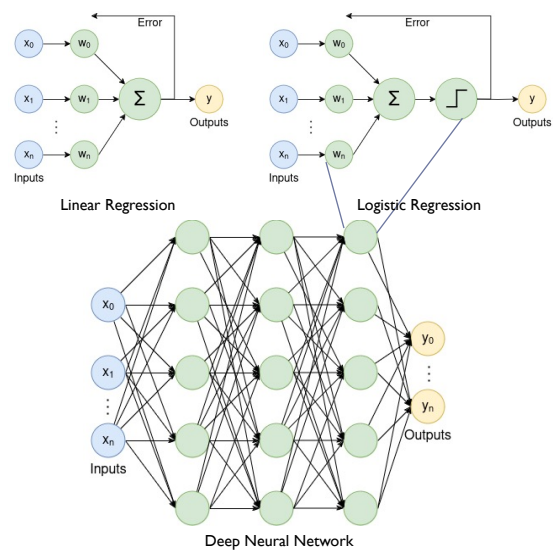
- A neural network maps from input neurons to output labels

- Each neuron performs logistic regression

$$h_{w,b}(x) = f(\omega^T x + b)$$

- Each input neuron represents a piece of the data (image pixel, transistor feature, net feature, ...)

- Works well for tabular data



42

Circuits as Images

43

- Images represent structural information of the IC
- Sub-components of circuit are isolated as different image representations

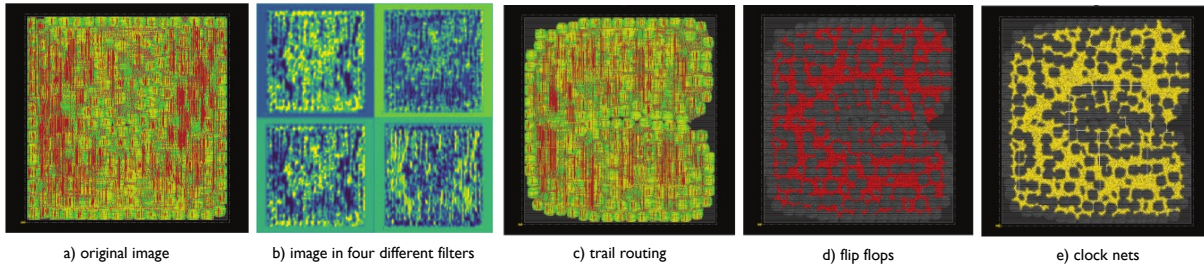



Image features

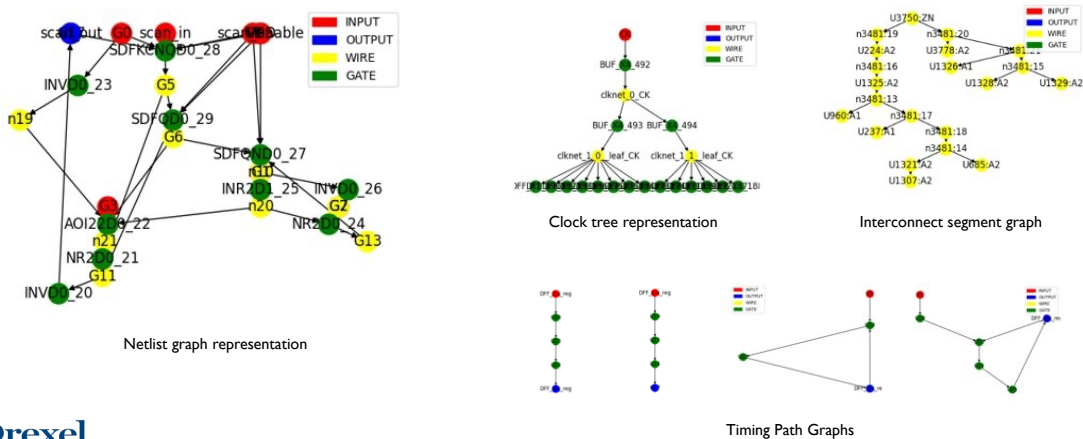
 Y.-C. Lu, J. Lee, A. Agnesina, K. Samadi, and S. K. Lim. "GAN-CTS: A generative adversarial framework for clock tree prediction and optimization," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1-8, Nov. 2019.

43

Circuits as Graph Representations

44

- Structural information and connections are represented efficiently as compared to an image
- Node and edge attributes as feature-set allows for a richer representation
- Positional information is lost

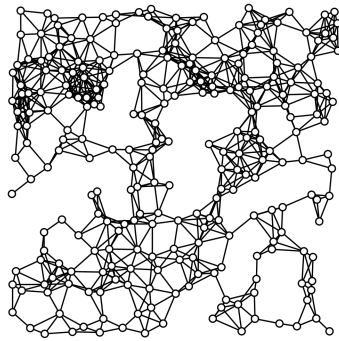


44

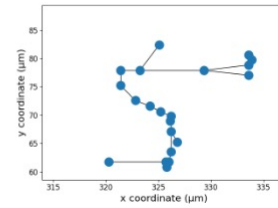
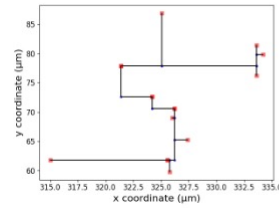
Spatial Graph Representations

45

- Contains structural and attribute information of graphs as well as positional information
- Spatial graph ML are new frontiers of research



Spatial graph network



Spatial Interconnect Graph



45

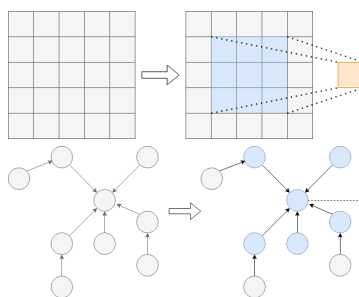
Convolution: Beyond Numerical Data

46

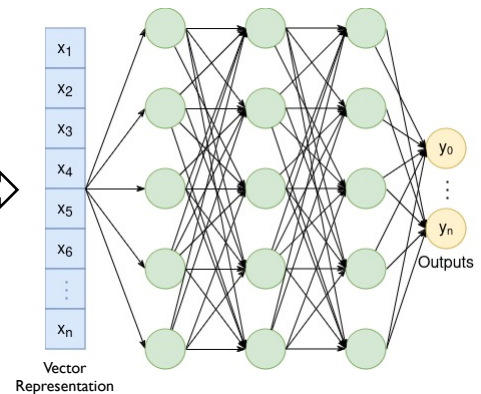
- **Convolutions**
 - Embed complex data structures into **lower-dimensional vector representations**
 - Combines local information through a series of learnable filters or aggregation functions
 - Embedded vectors can be used as inputs to a neural network

- Key data structures

- Images



Convolution

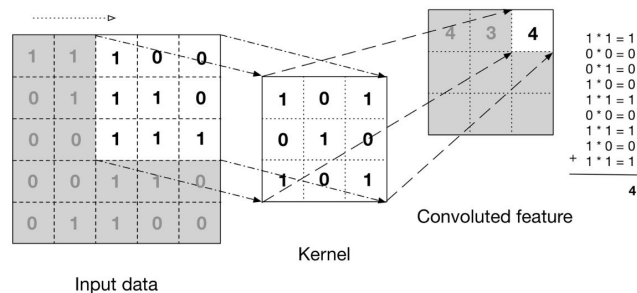


46

Convolutional Neural Network (CNN) Layer

47

- Operates on grid-like data structures where the data is represented in a structured, Euclidean space
- Each cell on the grid (pixel) is updated by the weighted sum of the current cell value and neighboring cell values
- Captures local patterns like edges, colors, and textures in the early layers, and more complex patterns (like parts of objects) in deeper layers



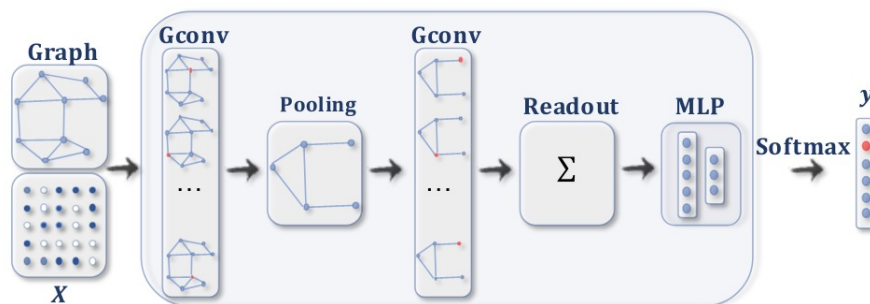
Source: analyticsvidhya.com

47

Graph Convolutional Neural Network (GCN) Layer

48

- Applied to graph-structured data, representing entities (nodes) and their relationships (edges) in a non-Euclidean space
- **Key idea:** Generate node embeddings based on **local network neighborhoods**
 - Converts graph structures and attributes into vector representations
 - Use CNN-like convolutional deep neural layers to train model



Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S.Y. Philip. "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 32, No. 1, pp. 4-24, Jan. 2020.

48

Vanilla Graph Convolutional Neural Networks (GCN)

49

Three primary steps:

- 1) Generate embeddings h_i for each graph node i
 - Map original feature vectors to vectors of a fixed dimension via linear neural network layers

Node embedding layers



49

Vanilla Graph Convolutional Neural Networks (GCN)

50

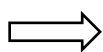
Three primary steps:

- 1) Generate embeddings h_i for each graph node i
 - Map original feature vectors to vectors of a fixed dimension via linear neural network layers
- 2) Aggregate embeddings of each node with embeddings of the neighboring nodes

$$h_i^{(l+1)} = \sigma(b^{(l)} + \sum_{j \in N(i)} \frac{1}{c_{ij}} W^{(l)} h_j^{(l)})$$

$h_i^{(l+1)}$: New embedding of node i
 σ : Nonlinear activation function
 $b^{(l)}$: Bias vector
 $N(i)$: Index set of direct neighboring nodes of node i
 $\frac{1}{c_{ij}}$: Normalization coefficient
 $W^{(l)}$: Weight vector
 $h_j^{(l)}$: Current embedding of node j

Node embedding layers



Aggregation layers



50

Vanilla Graph Convolutional Neural Networks (GCN)

51

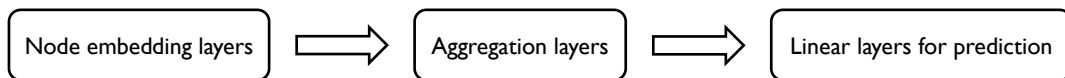
Three primary steps:

- 1) Generate embeddings h_i for each graph node i
 - Map original feature vectors to vectors of a fixed dimension via linear neural network layers
- 2) Aggregate embeddings of each node with embeddings of the neighboring nodes

$$h_i^{(l+1)} = \sigma(b^{(l)} + \sum_{j \in N(i)} \frac{1}{c_{ij}} W^{(l)} h_j^{(l)})$$

New embedding of node i Bias vector Index set of direct neighboring nodes of node i Normalization coefficient Weight vector Current embedding of node j

- 3) Send final embeddings to linear neural network layers for target predictions



51

Variants of Graph Neural Networks: GraphSAGE

52

- GraphSAGE (Graph Sample and AggreGatE)
- Extends the idea of GCN by allowing for inductive learning
 - Generalizes to unseen nodes after being trained only on a subset of the graph
- Introduces a sampling technique to reduce the computational load
 - Instead of using all neighbor nodes, GraphSAGE samples a fixed number of neighbors and aggregates features from those neighbors
 - **More scalable and applicable to large or dynamic graphs**
- Supports different types of aggregation functions
 - Mean pooling, max pooling



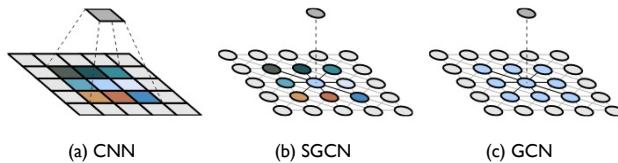
J. K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How Powerful are Graph Neural Networks?," *Proceedings of the International Conference on Learning Representations (ICLR)*, pp. 1–17, Dec 2018

52

Variants of Graph Neural Networks: SGCN

53

- Leverages node position
- Generalization of both GCNs and CNNs



Comparison of different neural convolutional filters. Each color denotes different trainable weights

- (a) Depicts convolutional filters for images
- (b) Depicts spatial graph convolutions, and
- (c) Depicts graph convolutions

Normal GCN

$$\hat{h}_i(U, b) = \sum_{j \in N_i} \text{ReLU}(b) h_j$$

Spatial GCN

$$\hat{h}_i(U, b) = \sum_{j \in N_i} \text{ReLU}(U^T(p_j - p_i) + b) \odot h_j$$

Node co-ordinates



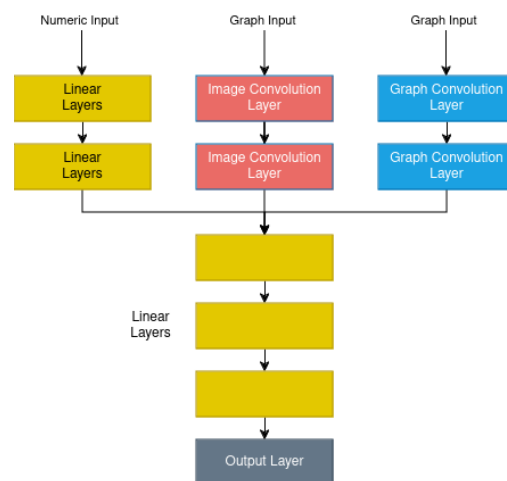
T. Danel, P. Spurek, J. Tabor, M. Smieja, Ł. Struski, A. Słowik, and Ł. Maziarka, "Spatial graph convolutional networks," *Proceedings of the International Conference on Neural Information Processing (NIPS)*, pp. 668–675, Nov. 2020

53

Multi-model Networks

54

- Integrate multiple types of data or modes (e.g., text, images, sound) into a single model
- Captures complex interactions not available from any single data type alone

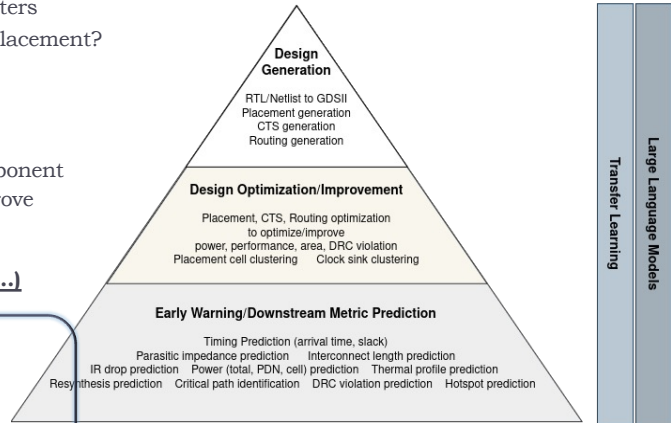


54

Machine Learning for EDA

55

- Prediction
 - Model the design space and predict circuit parameters
 - Can we predict post placement wirelength before placement?
 - **Regression, Classification, Clustering**
- Optimization
 - Improve on a circuit state or state of a circuit component
 - Can we use previous placement knowledge to improve current placed design?
 - **Clustering, Optimization (Gradient Descent, Simulate Annealing, Re-enforcement Learning ...)**
- Generation
 - Create a circuit component without any pre-existing state
 - Can we generate a completely new placement solution?
 - **RL/Generative ML**



55

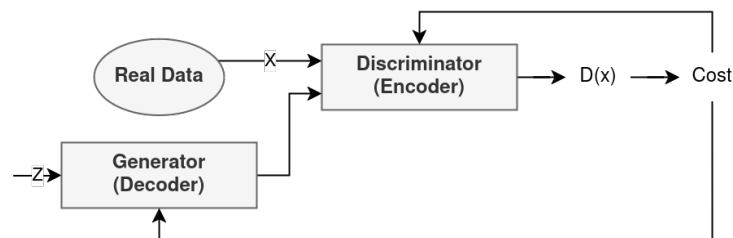
Generative Adversarial Networks

56

- Discriminator: $\max V(D) = \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]$

recognize real data recognize generated data
- Generator objective function: $\min V(G) = \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]$

optimize to fool discriminator
- Variants of GANs differ in objective function
- Suitable for prototype circuit design (layout) generation



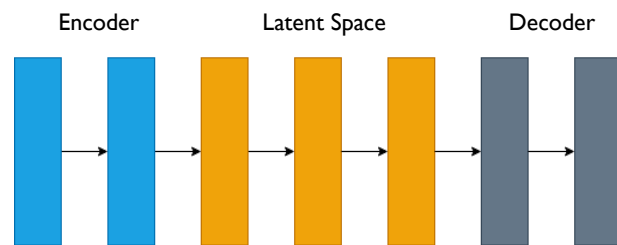
I. Goodfellow, et al., "Generative Adversarial Nets", *Proceedings of the International Conference on Neural Information Processing System (NIPS)*, Vol. 2, No. 1, pp.2672-2680, Dec. 2014

56

Variational Auto-Encoders (VAEs)

57

- Generative algorithm to encode distribution of training data, then generate new data with similar distribution
- Encoder: map input to a low-dimensional latent space
 - Effectively dimensionality reduction
- Decoder: convert signal in latent space back to input space
- Difference from GAN:
 - GAN generator takes noise as input
 - Higher-quality generation
 - Harder to train
 - VAE takes signal from the low-dimensional latent space as input
 - Lower-quality generation
 - Easier to train



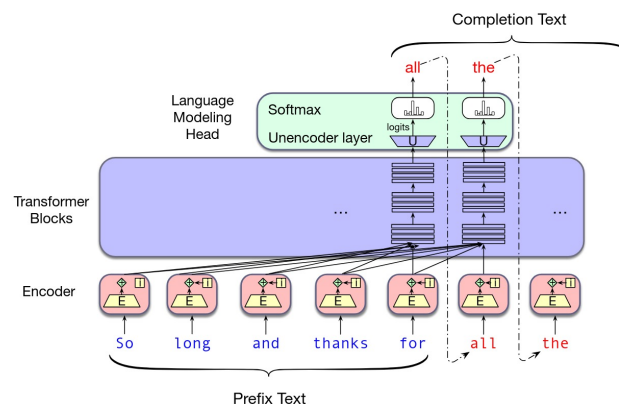
D. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *Proceedings of the International Conference on Learning Representations (ICLR)*, pp.1-14, Nov 2013

57

Large Language Models

58

- Transformer LLMs are next-token predictors
 - given a prefix text → model estimates probability distribution over possible next tokens
 - Model encodes prefix tokens → internal latent representation → then decoder + language modeling head selects the next token
 - Generation = repeating this step autoregressively token by token to build full output sequence



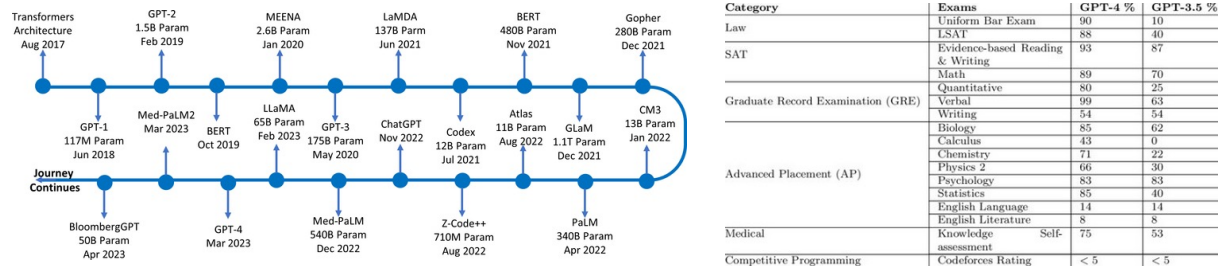
Source: web.stanford.edu/~jurafsky/slp3/slides/LLM24aug.pdf

58

LLMs across the years

59

- rapid scale increases (parameters + data + compute) has driven major capability jumps
- transition from simple text completion → reasoning-capable general foundation models
- current frontier models now outperform prior generations across law, STEM, medical, and standardized academic exam



S Mohamadi, G Mujtaba, N Le, G Doretto, DA.Adjeroh, "ChatGPT in the Age of Generative AI and Large Language Models: A Concise Survey", arXiv preprint arXiv:2307.04251, pp.1-60, July 2023

59

Outline of Presentation

60

- Introduction to Electronic Design Automation
- Machine learning techniques
- Case studies
- Standardizing ML for digital EDA
- Conclusions



60

Case Studies

61

Case Study 1: Prediction of Downstream Circuit Metrics in Physical Design

- Timing Prediction, Interconnect Parasitic Prediction, Power Prediction

Case Study 2: Generation and Optimization of Circuits

- Automated Placement, Clock Network Synthesis, Routing

Case Study 3: Transfer Learning Approaches

Case Study 4: Large Language Models in Physical Design



61

Case Studies

62

Case Study 1: Prediction of Downstream Circuit Metrics in Physical Design

- Timing Prediction, Interconnect Parasitic Prediction, Power Prediction

Case Study 2: Generation and Optimization of Circuits

- Automated Placement, Clock Network Synthesis, Routing

Case Study 3: Transfer Learning Approaches

Case Study 4: Large Language Models in Physical Design

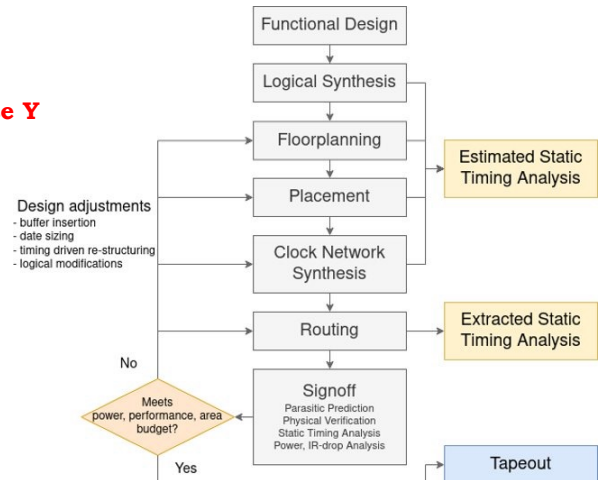


62

Case Study 1: Prediction of Downstream Circuit Metrics in Physical Design

63

- Can machine learning be used to predict downstream performance metrics?
 - Using information available in **initial phase X** predict the **performance metric P** in **final phase Y**
- Can the predictions be used as an **early warning system**?



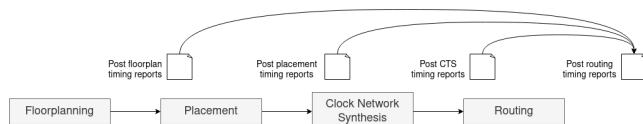
63

Problem 1: Timing Profile Prediction

64

Objective

- Predict the timing profile (arrival time, slack, critical path...) of the downstream stages of the physical design flow using the current stage



- Early and accurate prediction allows for pre-emptive timing optimization, reducing the need for costly iterations
- Improves circuit performance by identifying critical paths
- Minimizes design and verification costs and reduces the gap between estimated and actual timing

Startpoint: DFF_191/q_reg (rising edge triggered flip-flop clocked by CK)
Endpoint: DFF_186/q_reg (rising edge triggered flip-flop clocked by CK)
Last common point: CK
Path Group: CK
Path Type: max

Point	Fanout	Cap	Trans	Incr	Path
clock CK (rise edge)			0.00	0.00	0.00
clock network delay (propagated)			0.04	0.05	0.05
DFF_191/q_reg/CP (DF021)			0.04	0.00	0.05
DFF_191/q_reg/Q (DF001)			0.04	0.13	0.18
g059 (net)	1	0.00	0.04	0.00	0.18
U1246/I (IN006)			0.14	0.00	0.32
U1246/O (IN006)			0.14	0.00	0.32
g079 (net)	1	0.01	0.14	0.00	0.27
U1247/I (CN012)			0.35	0.25	0.52
U1247/O (CN012)			0.35	0.25	0.52
g1259 (net)	2	0.51	0.35	0.00	0.52
U1315/A2 (ND206)			0.14	0.14	0.66
U1315/O (ND206)			0.14	0.14	0.66
g1828 (net)	3	0.00	0.14	0.00	0.66
U1337/A3 (ND306)			0.06	0.00	0.74
U1337/O (ND306)			0.06	0.00	0.74
g1829 (net)	1	0.00	0.06	0.00	0.74
U1172/B (ND1206)			0.07	0.00	0.81
U1172/O (ND1206)			0.07	0.00	0.81
g1823 (net)	2	0.00	0.07	0.00	0.81
U1336/A2 (ND206)			0.10	0.00	0.89
U1336/O (ND206)			0.10	0.00	0.89
g1823 (net)	2	0.00	0.10	0.00	0.89
U1339/A1 (ND406)			0.09	0.00	0.89
U1339/O (ND406)			0.09	0.00	0.89
g1822 (net)	1	0.00	0.09	0.00	0.89
U1346/A2 (ND206)			0.09	0.00	0.98
U1346/O (ND206)			0.09	0.00	0.98
g1826 (net)	9	0.01	0.09	0.00	1.22
U1237/I (ND406)			0.25	0.00	1.22
U1237/O (ND406)			0.25	0.00	1.22
g1824 (net)	2	0.00	0.25	0.00	1.38
U1184/A2 (ND1206)			0.27	0.21	1.53
U1184/O (ND1206)			0.27	0.21	1.53
g1802 (net)	7	0.01	0.27	0.00	1.53
U1186/B (ND1206)			0.05	0.13	1.64
U1186/O (ND1206)			0.05	0.13	1.64
g1823 (net)	2	0.00	0.05	0.00	1.64
U1419/A1 (ND406)			0.05	0.00	1.64
U1419/O (ND406)			0.05	0.00	1.64
g1823 (net)	1	0.01	0.05	0.00	1.64
DFF_186/q_reg/Q (DF01)			0.20	0.00	1.82
data arrival time					1.82
clock CK (rise edge)			0.00	4.00	4.00
clock network delay (propagated)			0.05	4.05	4.05
clock reconvergence pessimism			0.00	4.05	4.05
DFF_186/q_reg/CP (DF01)			0.05	4.05	4.05
library setup time			-0.05	4.00	4.00
data required time					4.00
data arrival time					-1.82
slack (NET)					-2.10

Arrival Time

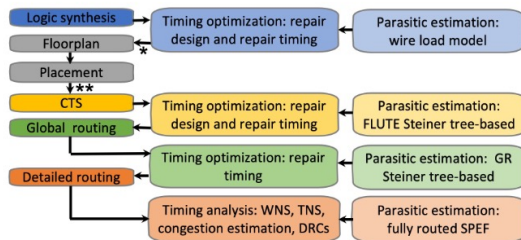


64

Post Global Route to Detail Route Timing Prediction

65

- Predict post-detailed routing wire delays and wire slews using post-global routing information
- Problem Formulation
 - Problem Type: Regression
 - Initial phase: Post global routing
 - Final phase: Post detailed routing
- Dataset
 - Circuits: Open-source designs
 - PDK: Open-source 45 nm and 130 nm
 - Toolset: OpenROAD



Design	Tech	# Nets	Post-DR WNS	
			GR-based parasitics	DR-based parasitics
DYNAMIC NODE	45nm	11598	-0.26ns	-0.26ns
AES	45nm	16836	-0.21ns	-0.19ns
IBEX	45nm	17566	-0.56ns	-0.60ns
JPEG	45nm	68247	-0.25ns	-0.17ns
RISCV32I	130nm	8150	-0.25ns	-0.17ns
IBEX	130nm	15307	-0.24ns	-0.11ns
AES	130nm	15369	-0.27ns	-0.09ns
JPEG	130nm	59573	0.24ns	0.02ns



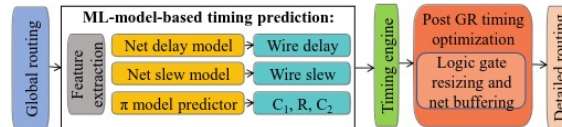
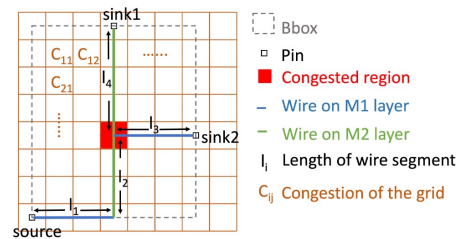
V. A. Chhabria, W. Jiang, A. B. Kahng and S. S. Sapatnekar, "From Global Route to Detailed Route: ML for Fast and Accurate Wire Parasitics and Timing Prediction," *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, pp. 7–14, Sept. 2022.

65

Post Global Route to Detail Route Timing Prediction

66

- Feature set (numeric)
 - HPWL
 - Number of sinks
 - Slew at the driving point
 - Congestion estimates
 - Rise and fall transitions
 - Source to sink length
 - Sources to sink R and C
- Model Architecture:
 - Three XGBoost models (classical ML)
 - Source-sink wire delay prediction mode
 - Source-sink wire slew prediction model
 - π -model parameter prediction model



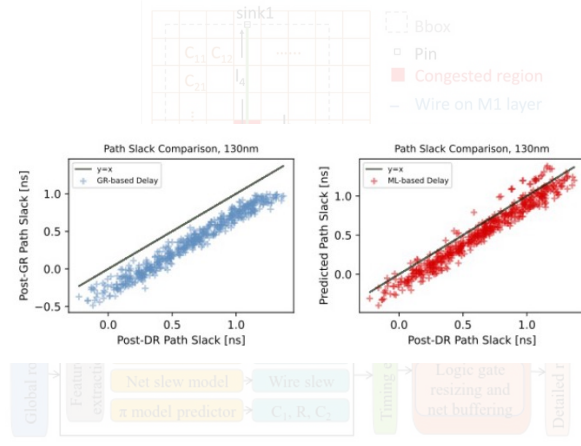
V. A. Chhabria, W. Jiang, A. B. Kahng and S. S. Sapatnekar, "From Global Route to Detailed Route: ML for Fast and Accurate Wire Parasitics and Timing Prediction," *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, pp. 7–14, Sept. 2022.

66

Post Global Route to Detail Route Timing Prediction

67

- Feature set (numeric)
 - HPWL
 - Number of sinks
 - Slew at the driving point
 - Congestion estimates
 - Rise and fall transitions
 - Source to sink length
 - Sources to sink R and C
- Model Architecture:
 - Three XGBoost models (classical ML)
 - Source-sink wire delay prediction mode
 - Source-sink wire slew prediction model
 - π -model parameter prediction model



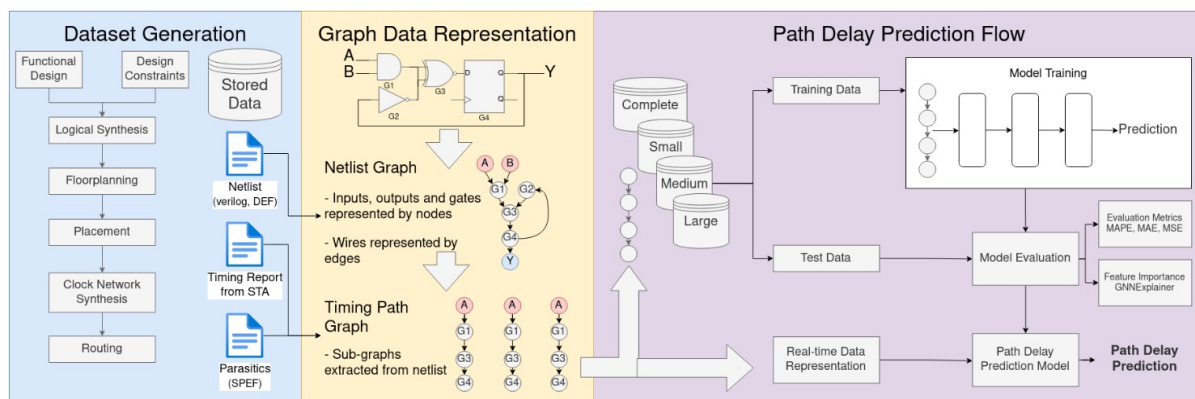
V. A. Chhabria, W. Jiang, A. B. Kahng and S. S. Sapatnekar, "From Global Route to Detailed Route: ML for Fast and Accurate Wire Parasitics and Timing Prediction," *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, pp. 7–14, Sept. 2022.

67

Arrival Time Prediction Framework

68

- Graph convolution based arrival time prediction
 - Initial Stage: Floorplan, Placement, CTS
 - Final Stage: Routing



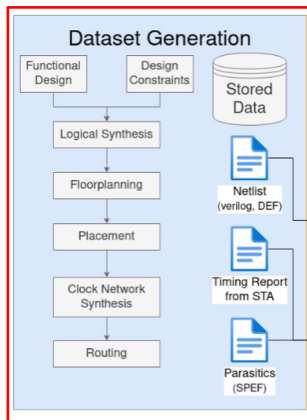
P. Shrestha, S. Phatharodom, and I. Savidis, "Graph representation learning for gate arrival time prediction," *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, pp. 127–133, Sept. 2022.

68

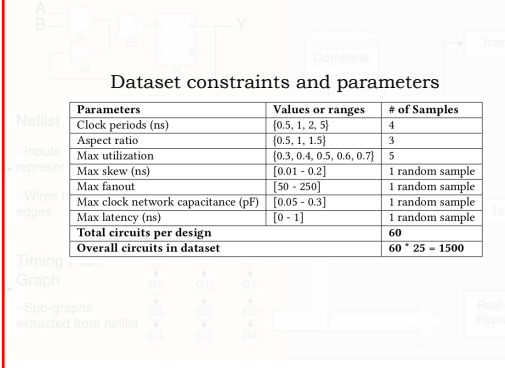
Arrival Time Prediction Framework: Dataset Generation

69

- Generate physical design data for each design stage
 - Multiple data files generated: Verilog file, DEF file, timing reports, SPEF file



Graph Data Representation



ISCAS'89 benchmark circuits

Dataset Group	Circuit	Circuit size	# of timing paths		
			Floorplan to routing	Placement to routing	CTS to routing
Small	s27	16	540	540	540
	s298	98	1027	1565	1566
	s344	136	1905	2310	2310
	s349	139	1885	2262	2262
	s382	129	1601	2571	2571
	s386	138	1195	1380	1380
	s400	136	1561	2644	2640
	s420	175	1175	1681	1681
	s444	149	1465	2687	2688
	s510	211	293	830	830
Medium	s641	185	3276	4283	4281
	s713	216	3138	4167	4167
	s820	298	1269	2186	2186
	s832	304	1439	2221	2222
	s838	355	2123	3284	3286
	s953	379	1796	3892	3892
	s1196	434	438	2678	2684
	s1238	474	538	2795	2797
	s1423	586	2753	6268	6309
	s9234	2313	8085	12525	12517
Large	s13207	3425	33174	47845	49092
	s15850	4209	21432	54959	55758
	s35932	14287	112861	221529	223327
	s38417	10479	33786	88409	92555
	s38584	13216	48423	142829	147508



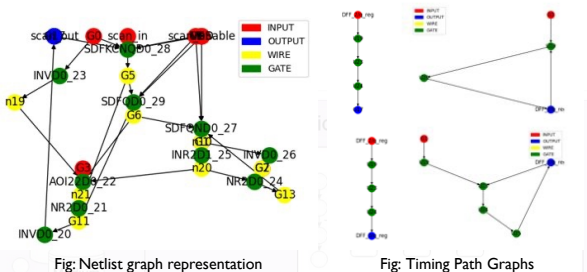
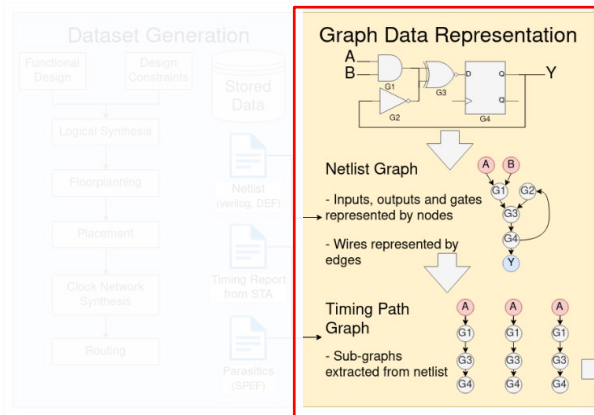
P. Shrestha, S. Phatharodom, and I. Savidis, "Graph representation learning for gate arrival time prediction," *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, pp. 127–133, Sept. 2022.

69

Arrival Time Prediction Framework: Graph Representation

70

- Utilize netlist and timing path reports
 - Convert into directed graphs
 - Populate with node features



Node features of timing path graphs

Feature Type	Feature	Size	Source
Setup features	Clock period	1	Design parameters / constraints
	Aspect ratio	1	
	Max utilization	1	
	Max skew	1	
	Max fan-out	1	
	Max clock network capacitance	1	
Standard cell features	Max latency	1	LEF/DEF file
	Standard cell one-hot encoding	859	
Structural features	Logic level	1	Calculated from netlist graph representation
	Number of fan-out gates	1	
Timing report features	Initial phase gate delay	1	STA timing report
	Initial phase arrival time	1	
Parasitic features	Total interconnect capacitance	1	IC Compiler generated SPEF file



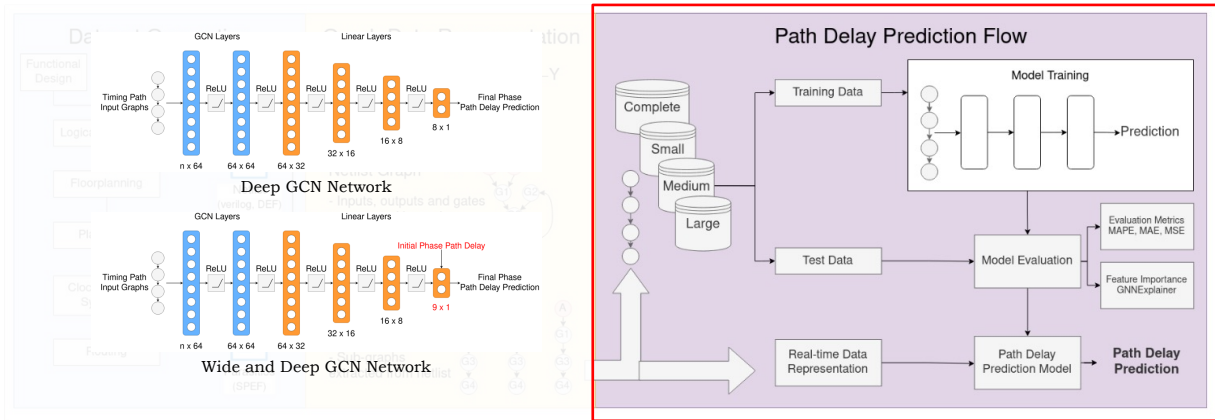
P. Shrestha, S. Phatharodom, and I. Savidis, "Graph representation learning for gate arrival time prediction," *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, pp. 127–133, Sept. 2022.

70

Arrival Time Prediction Framework: Model Training

71

- Graph convolutional layers for graph embedding
- Feed forward layers for network depth



P. Shrestha, S. Phatharodom, and I. Savidis, "Graph representation learning for gate arrival time prediction," *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, pp. 127–133, Sept. 2022.

71

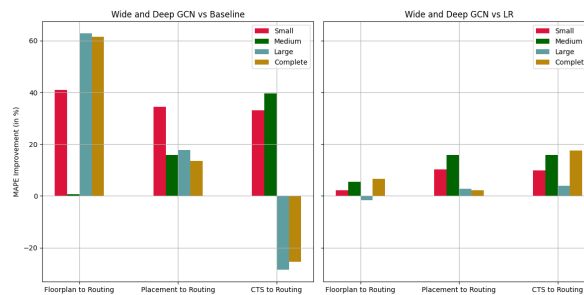
Results

72

- Wide and deep GCN
 - Outperforms the baseline for all scenarios except for CTS to routing timing estimate
 - Outperforms the baseline for CTS to routing timing estimate of small and medium circuits
 - Outperforms linear regression except for floorplan to routing timing estimate of large circuits
- Due to minimal changes in the timing paths after CTS, CTS to routing results in the best baseline performance
 - Hard to improve on result provided by tool post-CTS

MAPE and MAE comparisons

Dataset Group	Scenario	Baseline		Deep GCN		Wide and Deep GCN		LR	
		MAPE	MAE	MAPE	MAE	MAPE	MAE	MAPE	MAE
Complete	Floorplan to Routing	86.23	17.34	161.61	8.56	29.91	7.22	32	9.17
	Placement to Routing	49.41	14.77	47.34	15.53	39.94	10.51	40.81	13.88
	CTS to Routing	2.8	1.84	15.87	7.2	3.56	1.79	4.32	1.99
Small	Floorplan to Routing	7.25	1.94	13.35	3.07	4.28	1.06	4.37	1.12
	Placement to Routing	2.71	0.67	19.08	5	1.77	0.38	1.98	0.47
	CTS to Routing	2.69	0.66	23	6.55	1.8	0.39	2	0.48
Medium	Floorplan to Routing	6.46	2.68	35.35	9.81	6.42	1.6	6.79	1.69
	Placement to Routing	3.97	1.82	27.48	13.67	3.34	1.15	3.97	1.27
	CTS to Routing	3.9	1.8	22.2	7.86	2.35	1.12	2.79	1.22
Large	Floorplan to Routing	86.23	17.34	26.6	8.16	32.06	7.42	31.55	9.11
	Placement to Routing	49.41	14.77	33.31	11.43	40.61	10.36	41.74	14.01
	CTS to Routing	2.8	1.84	16.83	7.97	3.6	1.83	3.75	1.95



P. Shrestha, S. Phatharodom, and I. Savidis, "Graph representation learning for gate arrival time prediction," *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, pp. 127–133, Sept. 2022.

72

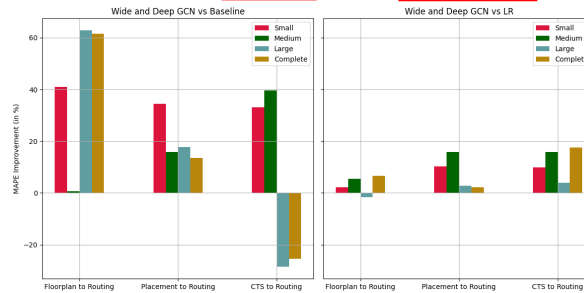
Results

73

- Wide and deep GCN
 - Outperforms the baseline for all scenarios except for CTS to routing timing estimate
 - Outperforms the baseline for CTS to routing timing estimate of small and medium circuits
 - Outperforms linear regression except for floorplan to routing timing estimate of large circuits
- Due to minimal changes in the timing paths after CTS, CTS to routing results in the best baseline performance
 - Hard to improve on result provided by tool post-CTS

MAPE and MAE comparisons

Dataset Group	Scenario	Baseline		Deep GCN		Wide and Deep GCN		LR	
		MAPE	MAE	MAPE	MAE	MAPE	MAE	MAPE	MAE
Complete	Floorplan to Routing	86.23	17.34	161.61	8.56	29.91	7.22	32	9.17
	Placement to Routing	49.41	14.77	47.34	15.53	39.94	10.51	40.81	13.88
	CTS to Routing	2.8	1.84	15.87	7.2	3.56	1.79	4.32	1.99
Small	Floorplan to Routing	7.25	1.94	13.35	3.07	4.28	1.06	4.37	1.12
	Placement to Routing	2.71	0.67	19.08	5	1.77	0.38	1.98	0.47
	CTS to Routing	2.69	0.66	23	6.55	1.8	0.39	2	0.48
Medium	Floorplan to Routing	6.46	2.68	35.35	9.81	6.42	1.6	6.79	1.69
	Placement to Routing	3.97	1.82	27.48	13.67	3.34	1.15	3.97	1.27
	CTS to Routing	3.9	1.8	22.2	7.86	2.35	1.12	2.79	1.22
Large	Floorplan to Routing	86.23	17.34	26.6	8.16	32.06	7.42	31.55	9.11
	Placement to Routing	49.41	14.77	33.31	11.43	40.61	10.36	41.74	14.01
	CTS to Routing	2.8	1.84	16.83	7.97	3.6	1.83	3.75	1.95



P. Shrestha, S. Phatharodom, and I. Savidis, "Graph representation learning for gate arrival time prediction," *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, pp. 127–133, Sept. 2022.

73

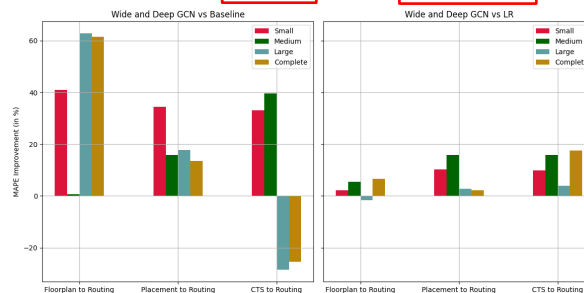
Results

74

- Wide and deep GCN
 - Outperforms the baseline for all scenarios except for CTS to routing timing estimate
 - Outperforms the baseline for CTS to routing timing estimate of small and medium circuits
 - Outperforms linear regression except for floorplan to routing timing estimate of large circuits
- Due to minimal changes in the timing paths after CTS, CTS to routing results in the best baseline performance
 - Hard to improve on result provided by tool post-CTS

MAPE and MAE comparisons

Dataset Group	Scenario	Baseline		Deep GCN		Wide and Deep GCN		LR	
		MAPE	MAE	MAPE	MAE	MAPE	MAE	MAPE	MAE
Complete	Floorplan to Routing	86.23	17.34	161.61	8.56	29.91	7.22	32	9.17
	Placement to Routing	49.41	14.77	47.34	15.53	39.94	10.51	40.81	13.88
	CTS to Routing	2.8	1.84	15.87	7.2	3.56	1.79	4.32	1.99
Small	Floorplan to Routing	7.25	1.94	13.35	3.07	4.28	1.06	4.37	1.12
	Placement to Routing	2.71	0.67	19.08	5	1.77	0.38	1.98	0.47
	CTS to Routing	2.69	0.66	23	6.55	1.8	0.39	2	0.48
Medium	Floorplan to Routing	6.46	2.68	35.35	9.81	6.42	1.6	6.79	1.69
	Placement to Routing	3.97	1.82	27.48	13.67	3.34	1.15	3.97	1.27
	CTS to Routing	3.9	1.8	22.2	7.86	2.35	1.12	2.79	1.22
Large	Floorplan to Routing	86.23	17.34	26.6	8.16	32.06	7.42	31.55	9.11
	Placement to Routing	49.41	14.77	33.31	11.43	40.61	10.36	41.74	14.01
	CTS to Routing	2.8	1.84	16.83	7.97	3.6	1.83	3.75	1.95



P. Shrestha, S. Phatharodom, and I. Savidis, "Graph representation learning for gate arrival time prediction," *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, pp. 127–133, Sept. 2022.

74

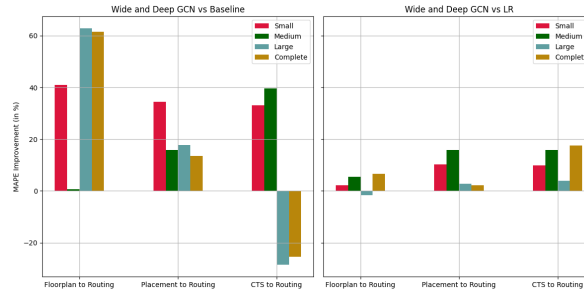
Results

75

- Wide and deep GCN
 - Outperforms the baseline for all scenarios except for CTS to routing timing estimate
 - Outperforms the baseline for CTS to routing timing estimate of small and medium circuits
 - **Outperforms linear regression except for floorplan to routing timing estimate of large circuits**
- Due to minimal changes in the timing paths after CTS, CTS to routing results in the best baseline performance
 - Hard to improve on result provided by tool post-CTS

MAPE and MAE comparisons

Dataset Group	Scenario	Baseline		Deep GCN		Wide and Deep GCN		LR	
		MAPE	MAE	MAPE	MAE	MAPE	MAE	MAPE	MAE
Complete	Floorplan to Routing	86.23	17.34	161.61	8.56	29.91	7.22	32	9.17
	Placement to Routing	49.41	14.77	47.34	15.53	39.94	10.51	40.81	13.88
	CTS to Routing	2.8	1.84	15.87	7.2	3.56	1.79	4.32	1.99
Small	Floorplan to Routing	7.25	1.94	13.35	3.07	4.28	1.06	4.37	1.12
	Placement to Routing	2.71	0.67	19.08	5	1.77	0.38	1.98	0.47
	CTS to Routing	2.69	0.66	23	6.55	1.8	0.39	2	0.48
Medium	Floorplan to Routing	6.46	2.68	35.35	9.81	6.42	1.6	6.79	1.69
	Placement to Routing	3.97	1.82	27.48	13.67	3.34	1.15	3.97	1.27
	CTS to Routing	3.9	1.8	22.2	7.86	2.35	1.12	2.79	1.22
Large	Floorplan to Routing	86.23	17.34	26.6	8.16	32.06	7.42	31.55	9.11
	Placement to Routing	49.41	14.77	33.31	11.43	40.61	10.36	41.74	14.01
	CTS to Routing	2.8	1.84	16.83	7.97	3.6	1.83	3.75	1.95



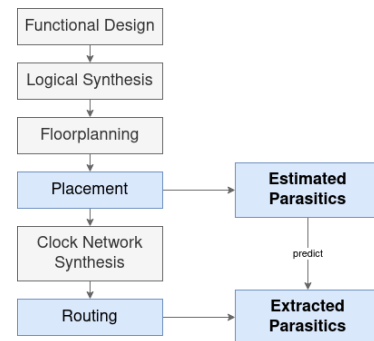
P. Shrestha, S. Phatharodom, and I. Savidis, "Graph representation learning for gate arrival time prediction," *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, pp. 127–133, Sept. 2022.

75

Problem 2: Interconnect Parasitic Impedance Prediction

76

- Interconnect impedance (R , L , C) are extracted after routing
 - Big impact on circuit performance
 - Need estimation of interconnect impedance at pre-routing design stages
- Early impedance used for estimation of circuit properties
 - Reduce error between intermediate stages and final stage simulation results of other circuit performance parameters
 - signal integrity
 - power profile
 - timing profile
 - gain
 - bandwidth
 - ...
 - Guide placement and routing
- Analytical models are not accurate enough
 - Solution: apply ML



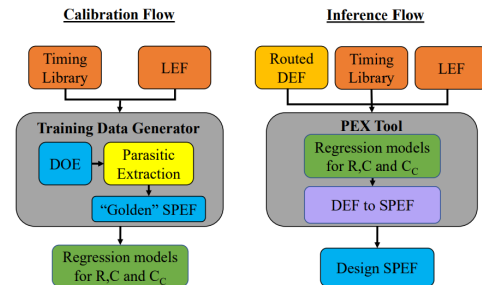
76

A Machine Learning Based Parasitic Extraction Tool

77

- Algorithm: regression
- Target circuit parameters for prediction: resistance, capacitance to ground, coupling, crossover, and cross-under capacitance of a net
- Training data is generated from Cadence Innovus with design of experiment (DOE)
 - Not exclusively for analog but provides physical modeling of interconnect capacitances
- Regression function is fixed
 - Only fitting regression parameters on data
 - Example: coupling capacitance expression

$$C_c = c/s + d \cdot l_{overlap} + e \cdot l_{overlap}/s + f$$
 - Inflexible to model interconnects at advanced technologies
- No results reported



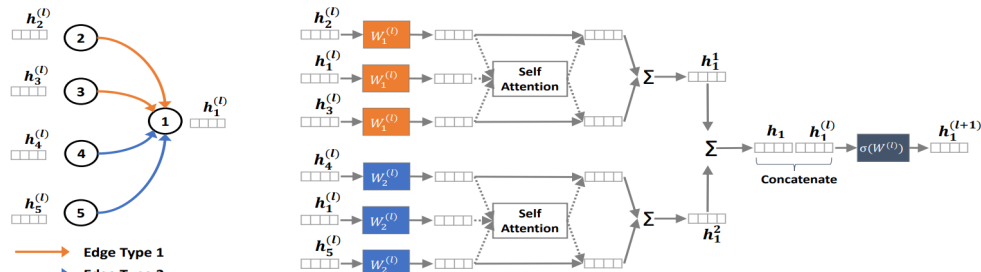
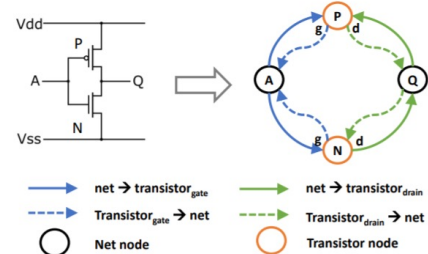
G. Pradipta, V. A. Chhabria, and S. S. Sapatnekar, "A Machine Learning Based Parasitic Extraction Tool," *Workshop on Open-Source EDA Technology (WOSET)*, pp. 1–3, Nov. 2019

77

ParaGraph: Layout Parasitics and Device Parameter Prediction with GNNs

78

- Graph representation of a circuit
 - Heterogeneous graph: devices and nets both as graph nodes
 - Multiple sub-models for different capacitance ranges
- Transistor features:
 - Gate poly length
 - Number of fingers
 - Number of fins
 - Multiplier



H. Ren, G. F. Kokai, W. J. Turner and T. Ku, "ParaGraph: Layout Parasitics And Device Parameter Prediction Using Graph Neural Networks," *Proceedings of the ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, June 2020.

78

ParaGraph: Layout Parasitics and Device Parameter Prediction with GNNs

79

- Simulation errors between pre-layout predictions and post-layout extracted values on 67 circuit metrics in the testing circuits

Error Range	Layout w/o parasitics	Designer's Estimation	Prediction w/ XGB	Prediction w/ ParaGraph
< 10%	4	6	17	44
10%-20%	0	17	14	10
20%-30%	5	18	4	8
30%-40%	35	2	7	4
40%-50%	14	6	9	1
> 50%	9	18	16	0
Mean	37.75%	>100%	32.14%	9.60%
Geometric Mean	29.01%	43.57%	15.46%	4.00%

- GCN-based model achieves an average prediction R2 of 0.772 (110% better than XGBoost)
- Average simulation errors from over 100% with designer's estimation to less than 10%

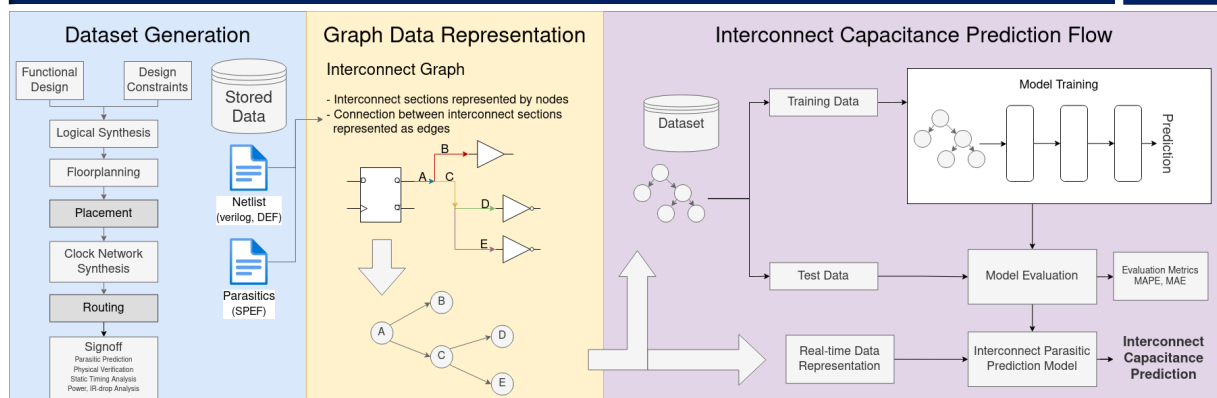


H. Ren, G. F. Kokai, W. J. Turner and T. Ku. "ParaGraph: Layout Parasitics And Device Parameter Prediction Using Graph Neural Networks," *Proceedings of the ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, June 2020.

79

Digital Parasitic Prediction Framework

80



- Objective**
 - Train GNN models for estimation of parasitic values on interconnect segments
 - Predict post routing capacitance using post placement circuit features

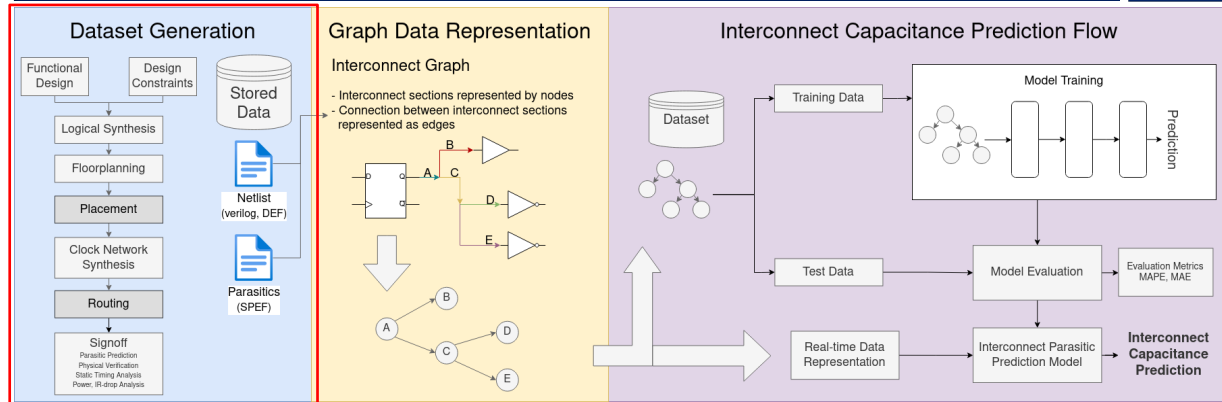


P. Shrestha and I. Savidis, "Graph Representation Learning for Parasitic Impedance Prediction of the Interconnect" *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, May 2023.

80

Parasitic Prediction Framework: Dataset Generation

81



- Generate physical design data for each design stage
 - Multiple data files generated: Verilog file, DEF file, SPEF file

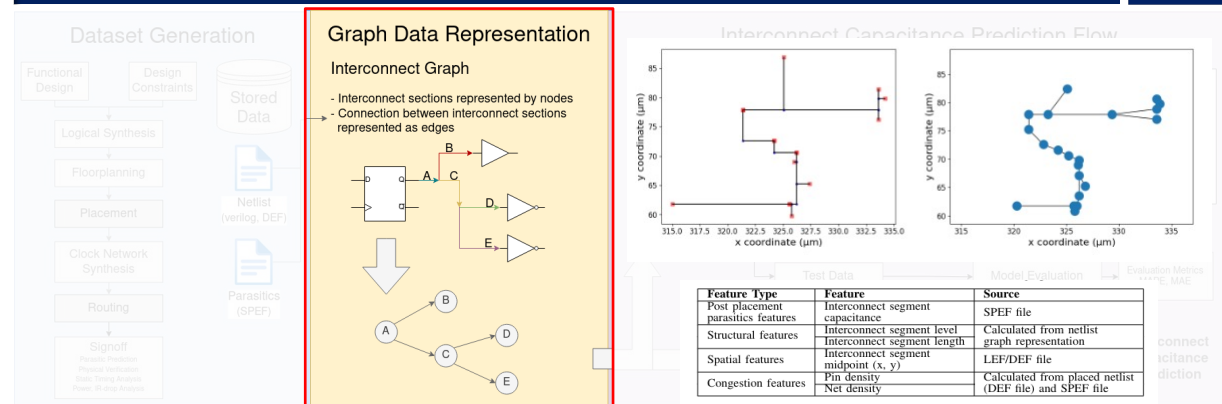


P. Shrestha and I. Savidis, "Graph Representation Learning for Parasitic Impedance Prediction of the Interconnect" *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, May 2023.

81

Parasitic Prediction Framework: Graph Representation

82



- Utilize netlist and SPEF reports
 - Convert into interconnect spatial graphs
 - Populate with node features

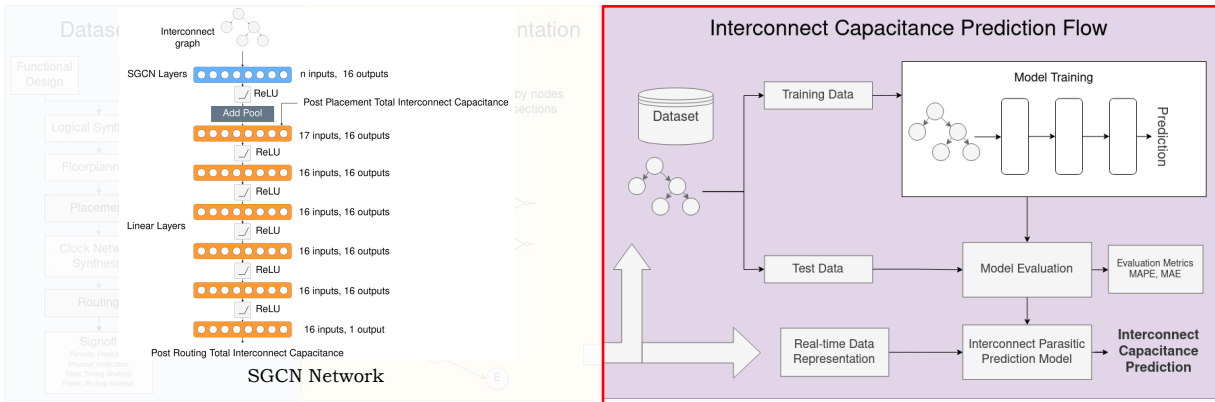


P. Shrestha and I. Savidis, "Graph Representation Learning for Parasitic Impedance Prediction of the Interconnect" *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, May 2023.

82

Parasitic Prediction Framework: Graph Representation

83



- Utilize Spatial GCN network model
 - Train 6 models where for each model one of the circuits is for test and the rest form the train set
 - Use mean square error (MSE) as loss optimization function



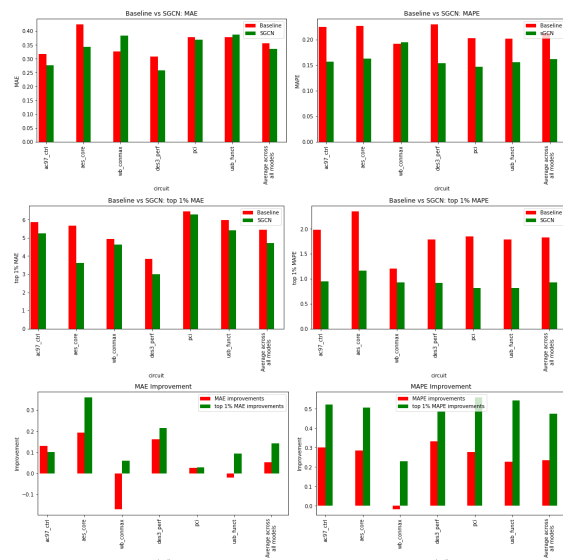
P. Shrestha and I. Savidis, "Graph Representation Learning for Parasitic Impedance Prediction of the Interconnect" *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, May 2023.

83

Results

84

- Considering total error
 - Model outperforms the baseline on average but not for all circuit scenarios
- Considering top 1% worse errors
 - Model outperforms the baseline for all scenarios consistently
- Average improvements
 - MAE: 5.33%
 - Top 1% MAE: 14.31%
 - MAPE: 23.39%
 - Top 1% MAPE: 47.43%
- R^2 across all models is consistent (> 0.95)



P. Shrestha and I. Savidis, "Graph Representation Learning for Parasitic Impedance Prediction of the Interconnect" *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, May 2023.

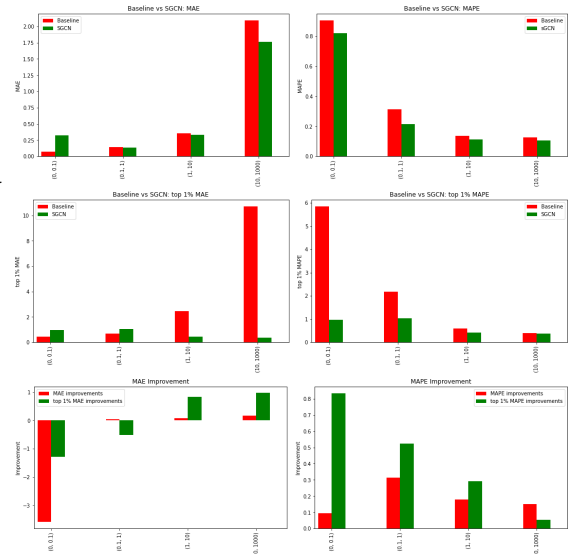
84

Results: interconnects distributed by capacitance range

85

- Considering MAE
 - Proposed model underperforms for smaller capacitances and outperforms for larger capacitances
- Considering MAPE
 - Proposed model outperforms the baseline consistently
- Large proportion of baseline error comes from nets with larger capacitance**
 - Model improving the MAE for larger nets is desirable**

Capacitance Range (fF)	No. of nets	Baseline	Proposed Model	Improvement
0 to 0.1	403	28.43	130.51	-359.09%
0.1 to 1	48533	6,808.11	6,503.50	4.47%
1 to 10	53104	18,908.16	17,504.80	7.42%
10 and beyond	4873	10,197.49	8,591.97	15.74%
Total	106913	35,942.19	32,730.77	8.93%



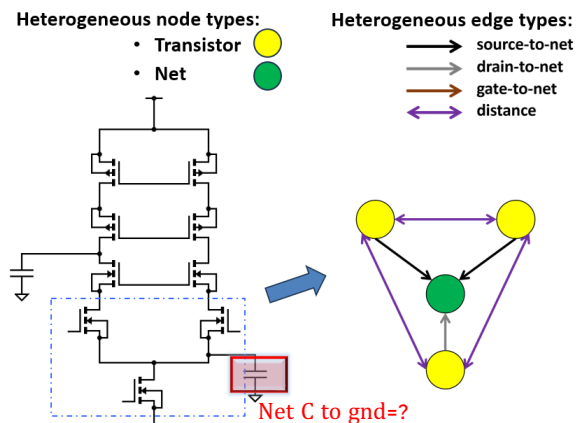
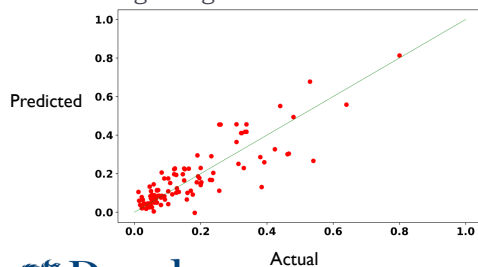
P. Shrestha and I. Savidis, "Graph Representation Learning for Parasitic Impedance Prediction of the Interconnect" *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, May 2023.

85

Analog Parasitic Prediction Framework

86

- Apply edge-weighted GNNs for estimation of interconnect capacitance
 - Schematic level
 - Post-placement level
- Post-placement model leverages coordinates of placed devices
 - Euclidean distance between devices used as edge weights



86

Problem 3: Total Power Prediction

87

- Early power prediction influences placement and routing decisions
- Crucial for optimizing energy efficiency and ensuring thermal management in electronic devices
- Key factors influencing power consumption
 - Static Power: Power consumed when the device is inactive but powered on
 - Dynamic Power: Power consumed in response to circuit activity
Mainly due to charging and discharging of capacitors

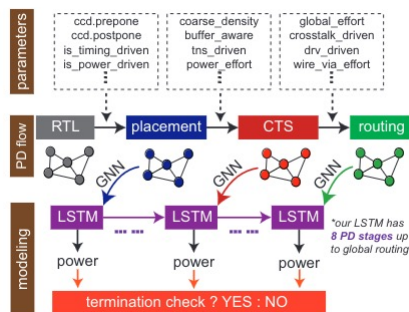


87

GNN + LSTM based Total Power Prediction

88

- Predict total power at the end of physical design flow in early physical design (PD) stages
- Problem Formulation
 - Problem Type: Regression
 - Initial phase: Post placement
 - Final phase: Post detailed routing
- Dataset
 - Circuits:
 - Two commercial CPU designs
 - Five OpenCore circuits
 - PDK: TSMC 28nm technology
 - Toolset: Synopsys DC Compiler, ICC2 Compiler
 - Parameters: 19 tool parameters influencing various physical design stages



Design Name	# Nets	# FFs	# Cells	Usage
CPU-A	206,224	22,366	202,791	training
ECG	85,058	14,018	84,127	
VGA	56,279	17,054	56,194	
JPEG	231,934	37,642	219,064	
CPU-B	542,391	47,552	597,085	testing
AES	90,905	10,688	113,168	
LDPC	42,018	2,048	39,377	



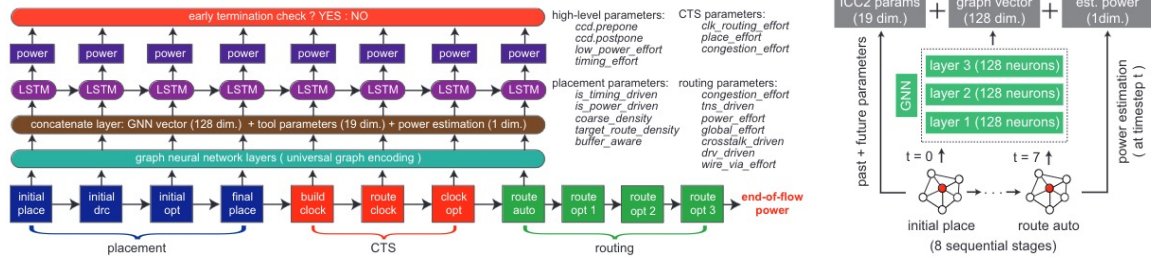
Y.-C. Lu, W.-T. Chan, V. Khandelwal, and S. K. Lim, "Driving Early Physical Synthesis Exploration through End-of-Flow Total Power Prediction," *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, pp. 97–102, Sept. 2022.

88

GNN + LSTM based Total Power Prediction

89

- Feature set
 - Minimum, maximum slack
 - Maximum transition of input/output pins
 - Switching power of driving net
 - Cell power (switching, internal, leakage)
- Model Architecture
 - GNN for netlist representation learning
 - Long Short Term Memory (LSTM) for sequential modeling of design stages



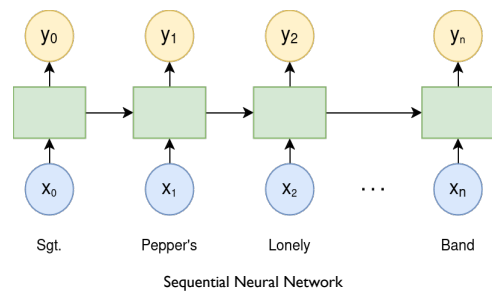
Y.-C. Lu, W.-T. Chan, V. Khandelwal, and S. K. Lim, "Driving Early Physical Synthesis Exploration through End-of-Flow Total Power Prediction," *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, pp. 97–102, Sept. 2022.

89

Long Short Term Memory (LSTM)

90

- Recurrent Neural Network (RNN)
 - **Encodes sequential data into vector representation**
 - Does not capture long term dependencies
- LSTM extends RNN
 - Incorporates long-term memory components to prioritize retaining certain hidden states over others.
 - Special units (memory cells) maintain state over time
 - Forget Gate:
Decides what information to discard from the cell state.
 - Input Gate:
Determines which values from the input to update the cell state.
 - Output Gate:
Controls the output and the next hidden state.



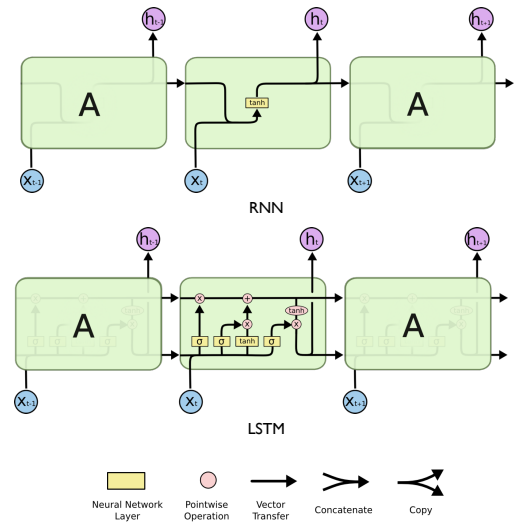
Source: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

90

Long Short Term Memory (LSTM)

91

- Recurrent Neural Network (RNN)
 - Encodes sequential data into vector representation
 - Does not capture long term dependencies
- LSTM extends RNN
 - Incorporates long-term memory components to prioritize retaining certain hidden states over others.
 - Special units (memory cells) maintain state over time
 - Forget Gate:
Decides what information to discard from the cell state.
 - Input Gate:
Determines which values from the input to update the cell state.
 - Output Gate:
Controls the output and the next hidden state.



Source: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>



91

GNN + LSTM based Total Power Prediction

92

- Feature set
 - Minimum, maximum slack
 - Maximum transition of input/output pins
 - Switching power of driving net
 - Cell power (switching, internal, leakage)
- Model Architecture
 - GNN for netlist representation learning
 - LSTM for sequential modeling of design stages

CC denotes the Pearson correlation coefficient.
NRMSE denotes the accuracy of the proposed model
All metrics are computed against ICC2 power values.

PD stage (avg time)	unseen design	NRMSE (%)	ICC2 CC	our CC
initial place (3%)	CPU-B	29.8	0.42	0.46
	AES	24.7	0.26	0.5
	LDPC	21.2	0.18	0.37
initial drc (4%)	CPU-B	22.1	0.43	0.58
	AES	28.6	0.25	0.52
	LDPC	27.3	0.18	0.38
initial opt (7%)	CPU-B	18.5	0.42	0.72
	AES	12.1	0.32	0.68
	LDPC	12.9	0.31	0.66
final place (22%)	CPU-B	11.2	0.45	0.81
	AES	9.7	0.35	0.86
	LDPC	9.2	0.32	0.72
build clock (6%)	CPU-B	8.2	0.41	0.89
	AES	7.1	0.47	0.9
	LDPC	8.7	0.43	0.88
route clock (7%)	CPU-B	5.9	0.42	0.94
	AES	6.4	0.76	0.92
	LDPC	5.8	0.74	0.93
clock opt (12%)	CPU-B	5.2	0.65	0.95
	AES	6.4	0.96	0.96
	LDPC	3.9	0.92	0.95
route auto (8%)	CPU-B	4.1	0.75	0.98
	AES	5.3	0.96	0.97
	LDPC	3.7	0.94	0.97



Y.-C. Lu, W.-T. Chan, V. Khandelwal, and S. K. Lim, "Driving Early Physical Synthesis Exploration through End-of-Flow Total Power Prediction," *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, pp. 97–102, Sept. 2022.

92

Case Studies

93

Case Study 1: Prediction of Downstream Circuit Metrics in Physical Design

- Timing Prediction, Interconnect Parasitic Prediction, Power Prediction

Case Study 2: Generation and Optimization of Circuits

- Automated Placement, Clock Network Synthesis, Routing

Case Study 3: Transfer learning approaches

Case Study 4: Large Language Models in Physical Design



93

Case Study 2: Design Optimization and Generation

94

- Design Optimization
 - Given an **initial circuit component state S** , can we achieve a **final circuit component state S'** to improve the **performance metric P** of the said circuit component state?
- Generation vs Optimization
 - Can previous-stage circuit component state act as the initial stage for design generation?
- ML for design optimization and generation
 - Reinforcement learning
 - Generative learning



94

Placement, CTS, and Routing for Digital Design

95

- Primary Objectives
 - **Performance Optimization:** Tailoring chip layout to maximize circuit performance
 - **Power and Area Efficiency:** Reducing power consumption and minimizing chip area, often through wirelength minimization
 - **Design Implementation:** Ensuring the design is reliable, scalable, and completed within project timelines
- Design Rule Constraints
 - Adhering to stringent design rules that govern the placement of circuit elements and routing to ensure manufacturability and functional integrity of the chip
- Additional Objectives
 - Low congestion
 - Thermal Management
 - Noise Reduction

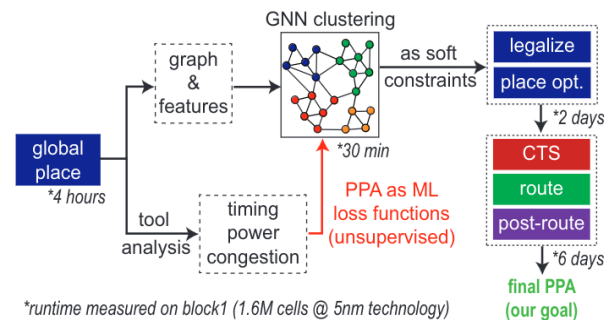


95

Digital Placement Optimization via PPA-Directed Graph Clustering

96

- Optimize power, performance, and area (PPA) during circuit placement
- Graph based cell clustering
 - Provide clusters as inputs to commercial placers
- Use post placement congestion, timing, and power to optimize cluster



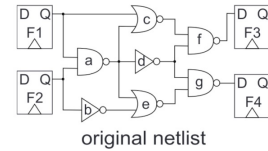
Y.-C. Lu, T. Yang, S. K. Lim, and H. Ren, "Placement optimization via PPA-directed graph clustering", *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, pp. 1–6, Sept. 2022

96

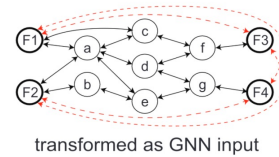
Netlist Representation

97

- Utilized netlist graphs of circuits
 - Post-placement node-specific features with physical, timing, power attributes
 - "Skip-connections" to aid GNN model in capturing timing-related attributes



Type	# dim.	Description
name embeddings	16	hierarchical name encoded by S-BERT [11]
memory affinity	M	shortest logic distance to each memory
worst output slack	1	worst slack value at output pin
worst output slew	1	maximum transition at output pin
worst input slew	1	maximum transition among input pin(s)
largest activity	1	largest switching activity value among nets
locations	2	(x,y) location of initial placement



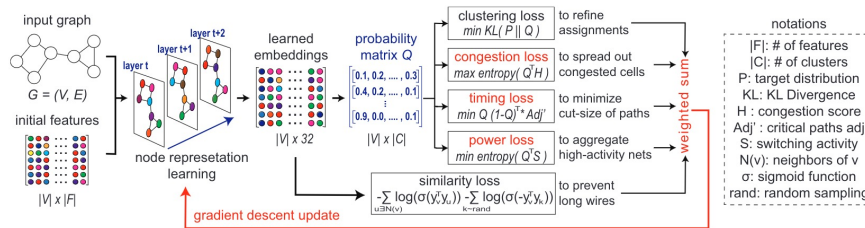
Y.-C. Lu, T. Yang, S. K. Lim, and H. Ren, "Placement optimization via PPA-directed graph clustering", *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, pp. 1–6, Sept. 2022.

97

Deep Graph Clustering

98

- Generate graph embeddings using GraphSAGE
- Obtain initial cluster centroids/clustering assignments using K-means



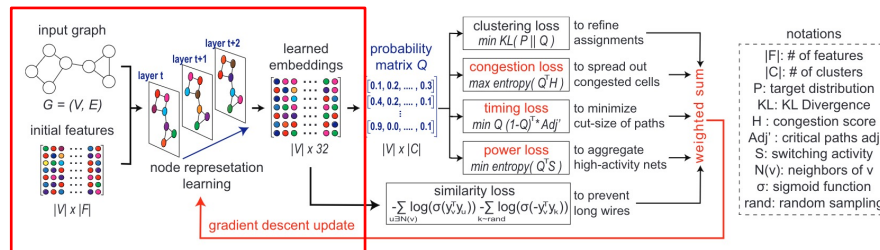
Y.-C. Lu, T. Yang, S. K. Lim, and H. Ren, "Placement optimization via PPA-directed graph clustering", *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, pp. 1–6, Sept. 2022.

98

Deep Graph Clustering

99

- Generate graph embeddings using GraphSAGE
- Obtain initial cluster centroids/clustering assignments using K-means



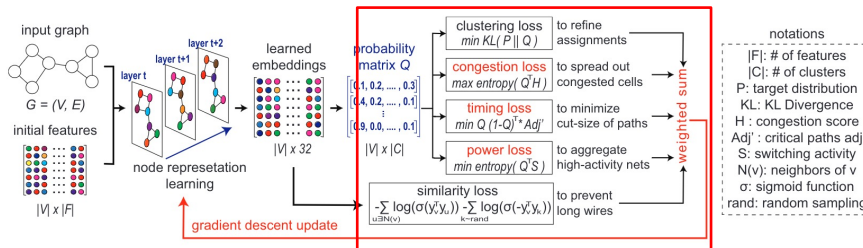
Y.-C. Lu, T. Yang, S. K. Lim, and H. Ren, "Placement optimization via PPA-directed graph clustering", *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, pp. 1–6, Sept. 2022.

99

Deep Graph Clustering

100

- Generate graph embeddings using GraphSAGE
- Obtain initial cluster centroids/clustering assignments using K-means
- Optimize the cluster centroids/clustering assignments by minimizing loss functions
 - Clustering loss: Kullback-Leibler divergence
 - Congestion loss: maximizing Shannon entropy
 - Power loss: minimizing entropy of maximum switching activities
 - Timing loss: optimizing clustering of cells on critical timing paths
 - Similarity loss: minimize embedding distance in high dimensions



Y.-C. Lu, T. Yang, S. K. Lim, and H. Ren, "Placement optimization via PPA-directed graph clustering", *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, pp. 1–6, Sept. 2022.

100

Deep Graph Clustering: Results

101

- PPA improvements for all benchmark circuits
 - 2.2% improvement in power consumption
 - 26% improvement in WNS
 - 1.4% improvement in wirelength
- Lower number of DRC violations

	design, # of clusters	design1, C =10		design2, C =11		design3, C =11		design4, C =10	
		post- place	post- route	post- place	post- route	post- place	post- route	post- place	post- route
default industrial PD flow (no clustering)	WNS (ps)	-48	-88	-13	-29	-4	-20	-5	-22
	TNS (ns)	-41.45	-2.89	-0.084	-3.82	-0.019	-2.24	-0.042	-2.65
	# of violations	3306	574	23	1440	10	996	16	1221
	total WL	1	1	1	1	1	1	1	1
	clock WL	-	1	-	1	-	1	-	1
	total power	1	1	1	1	1	1	1	1
	clock power	-	1	-	1	-	1	-	1
default + unsupervised clustering [11]	WNS (ps)	-39	-9	-4	-22	-1	-15	-2	-12
	TNS (ns)	-14.18 (-66%)	-0.32 (-88%)	-0.007 (-92%)	-3.07 (-20%)	-0.001 (-95%)	-1.62 (-28%)	-0.007 (-83%)	-2.16 (-19%)
	# of violations	1622	149	2	1288	1	823	4	1103
	total WL	0.999	1	1	0.998	0.998	0.996	0.977	0.976
	clock WL	-	0.994	-	0.999	-	0.978	-	0.985
	total power	1	1.001	1.001	0.999	0.997	0.996	0.992	0.991
	clock power	-	0.994	-	0.989	-	0.998	-	0.986



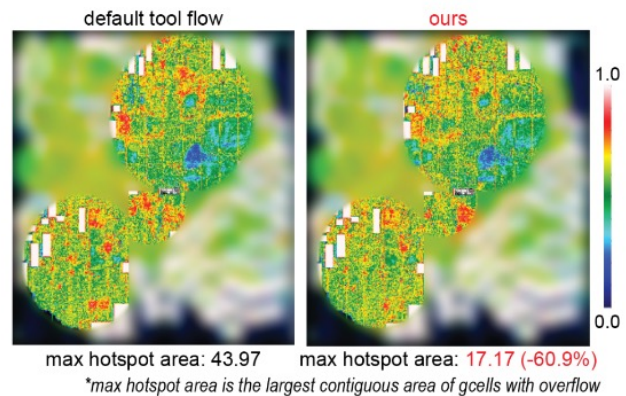
Y.-C. Lu, T. Yang, S. K. Lim, and H. Ren, "Placement optimization via PPA-directed graph clustering", *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, pp. 1-6, Sept. 2022

101

Deep Graph Clustering: Results

102

- PPA improvements for all benchmark circuits
 - 2.2% improvement in power consumption
 - 26% improvement in WNS
 - 1.4% improvement in wirelength
- Lower number of DRC violations
- Reduction of 60.9% in the routing congestion of hotspot locations



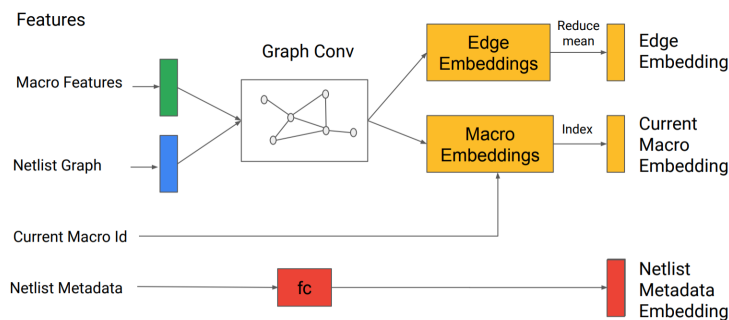
Y.-C. Lu, T. Yang, S. K. Lim, and H. Ren, "Placement optimization via PPA-directed graph clustering", *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, pp. 1-6, Sept. 2022

102

Automated Cell Placement By Google

103

- Primarily for digital cell placement optimization with RL and GNN
 - Applications: design google accelerator chips (TPUs)
- RL for placing macros and heuristics to place standard cells
 - RL reward: expected wirelength (i.e., HPWL) and expected congestion
- Edge-based GNN operate on embeddings of placed partial graph and candidate node



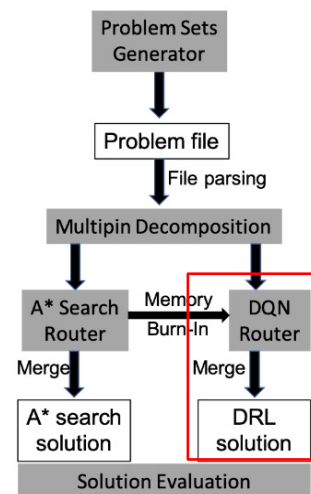
A. Mirhoseini, et al., "A Graph Placement Methodology for Fast Chip Design", *Nature*, No. 594, pp. 207–212, June 2021

103

Deep Re-Enforcement Learning for Global Routing

104

- Problem Formulation
 - Predict optimality of routing measured by overflow and minimization of total wire length
 - Initial phase: Placement
 - Final phase: Detailed routing
 - Baseline: Sequential A* algorithm
- Files utilized for RL global routing (Problem file)
 - Specify the dimensions of the routing grid (e.g., width, height, and layers)
- A Deep Q-network (DQN)



H. Liao, W. Zhang, X. Dong, B. Poczoz, K. Shimada, & L. B. Kara, "A deep reinforcement learning approach for global routing," *Journal of Mechanical Design*, Vol. 142, No. 6, pp 061701, Nov. 2019.

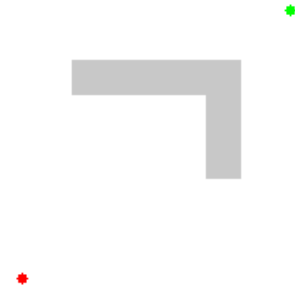
104

A* algorithm

105

- Routing regions represented as a graph
- Find shortest path between a given node and a target node
- Given an initial and final cell on a square grid
 - g : cost of moving from the initial cell to a certain cell on grid
 - h : estimated cost of moving from the current cell to the final cell
 - Euclidean distance
 - Manhattan distance
 - $f = g + h$
 - Procedure: select and move to the smallest f -valued cell
- Limitation
 - High space complexity as storage of all nodes in paths is required

A* search between bottom-left red dot to upper-right green dot



Source: Wikipedia

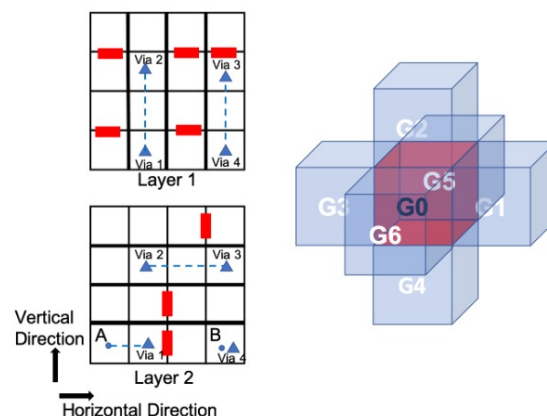


105

Deep Q-network (DQN) for Global Routing

106

- States Definition
 - x, y, z coordinates representing the agent's current position within the routing grid
- State Updates
 - Location Update: Position of the routing agent
 - Capacity Adjustment: Wire utilization



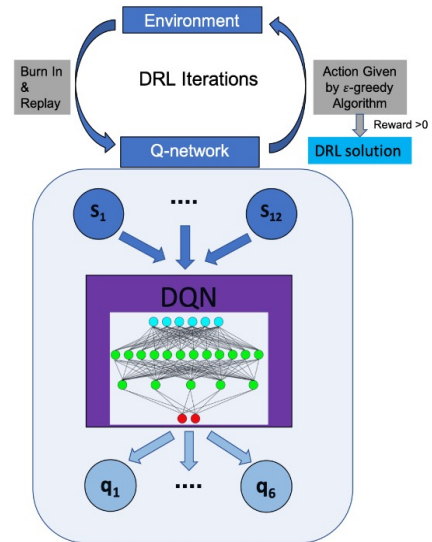
H. Liao, W. Zhang, X. Dong, B. Poczos, K. Shimada, & L. B. Kara, "A deep reinforcement learning approach for global routing," *Journal of Mechanical Design*, Vol. 142, No. 6, pp 061701, Nov. 2019.

106

Deep Q-network (DQN) for Global Routing

107

- States Definition
 - x, y, z coordinates representing the agent's current position within the routing grid.
 - State Updates
 - Location Update: Position of the routing agent
 - Capacity Adjustment: Wire utilization
- Network Architecture
 - Comprises several fully connected layers
 - Layers are followed by activation functions (ReLU)
- Reward Mechanism
 - Overflow (OF) and Total Wire Length (WL)



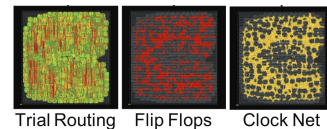
H. Liao, W. Zhang, X. Dong, B. Poczoz, K. Shimada, & L. B. Kara, "A deep reinforcement learning approach for global routing," *Journal of Mechanical Design*, Vol. 142, No. 6, pp 061701, Nov. 2019.

107

GAN-CTS: Generative Adversarial Learning for Clock Tree Optimization

108

- Problem Formulation
 - Objective: Predict optimal CTS parameter setup for a commercial tool
 - Baseline: Clock power, clock wirelength, and max skew of CTS generated on commercial tool's auto-setting
 - Input data: Post-placement layout images
- Dataset Generation
 - Toolset: Synopsys Design Compiler, Cadence Innovus
 - PDK: TSMC 28nm
 - No of CTS samples
 - Per design: $5 * 7 * 100 = 3500$
 - Total dataset: $3500 * 7 = 24500$
 - Train/test split
 - Training set: ARM, ECG, JPEG, LDPC, TATE
 - Testing set: AES, NOVA



Netlist Name	# Nets	# Flip Flops	# Total Cells
AES-128	90,905	10,688	113,168
Arm-Cortex-M0	13,267	1,334	12,942
NOVA	138,171	29,122	136,537
ECG	85,058	14,018	84,127
JPEG	231,934	37,642	219,064
LDPC	42,018	2,048	39,377
TATE	185,379	31,409	184,601

Type	Parameters	Values or ranges
Placement	aspect ratio	0.5, 0.75, 1.0, 1.25, 1.5
	utilization	0.4, 0.45, 0.5, ..., 0.7
	max skew (ns)	[0.01, 0.2]
CTS	max fanout	[50, 250]
	max cap trunk (pF)	[0.05, 0.3]
	max cap leaf (pF)	[0.05, 0.3]
	max slew trunk (ns)	[0.03, 0.3]
	max slew leaf (ns)	[0.03, 0.3]
	max latency (ns)	[0, 1]
	max early routing layer	2, 3, 4, 5, 6
	min early routing layer	1, 2, 3, 4, 5
	max buffer density	[0.3, 0.8]



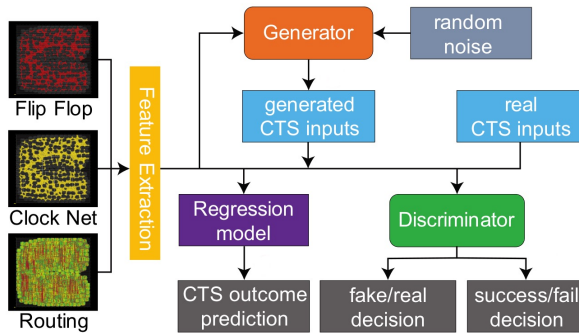
Reference: Y.-C. Lu, J. Lee, A. Agnesina, K. Samadi, and S. K. Lim, "GAN-CTS: A generative adversarial framework for clock tree prediction and optimization", Proceedings of the IEEE/ACM International Conference on Computer Aided Design (ICCAD), pp. 1-8, 2019

108

GAN-CTS Framework

109

- Use post-placement layout images to predict and optimize CTS parameters
- Consists of four separate models



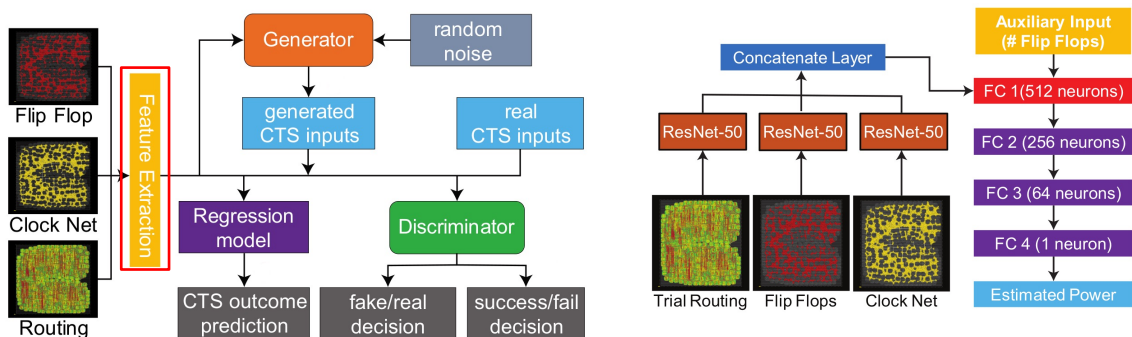
Y.-C. Lu, J. Lee, A. Agnesina, K. Samadi, and S. K. Lim, "GAN-CTS: A generative adversarial framework for clock tree prediction and optimization," *Proceedings of the IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 1–8, Nov 2019

109

GAN-CTS Framework: Placement Image Feature Extraction

110

- Use ResNet-50 (pre-trained deep image network) for feature extraction



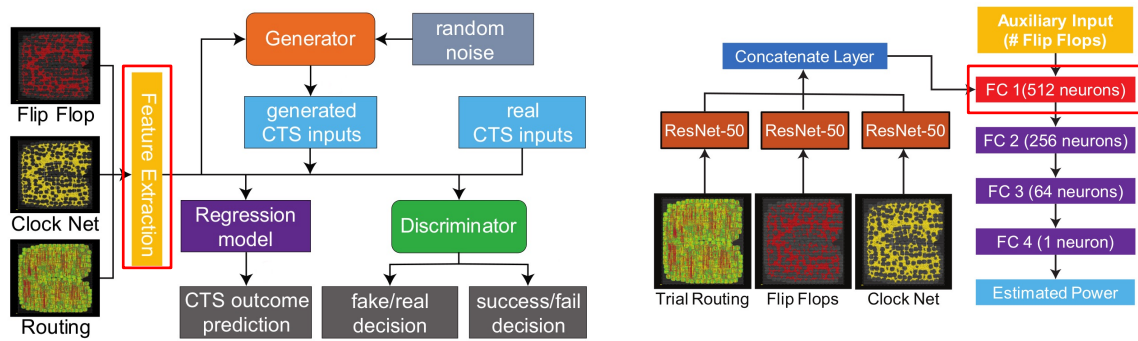
Y.-C. Lu, J. Lee, A. Agnesina, K. Samadi, and S. K. Lim, "GAN-CTS: A generative adversarial framework for clock tree prediction and optimization," *Proceedings of the IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 1–8, Nov 2019

110

GAN-CTS Framework: Placement Image Feature Extraction

111

- Use ResNet-50 (pre-trained deep image network) for feature extraction
- Embed placement images into low dimensional vector embeddings



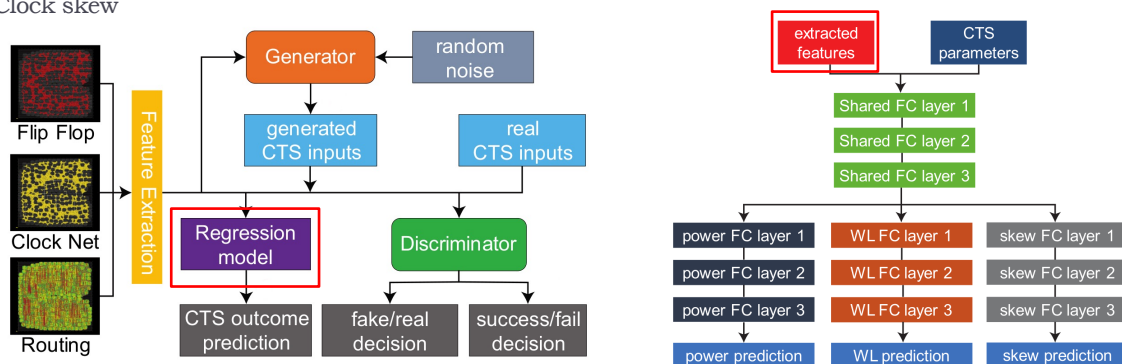
Y.-C. Lu, J. Lee, A. Agnesina, K. Samadi, and S. K. Lim, "GAN-CTS: A generative adversarial framework for clock tree prediction and optimization," *Proceedings of the IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 1–8, Nov 2019

111

GAN-CTS Framework: CTS Outcomes Prediction

112

- Use the extracted features and CTS parameters to predict for circuit properties
 - Power consumption
 - Wirelength
 - Clock skew



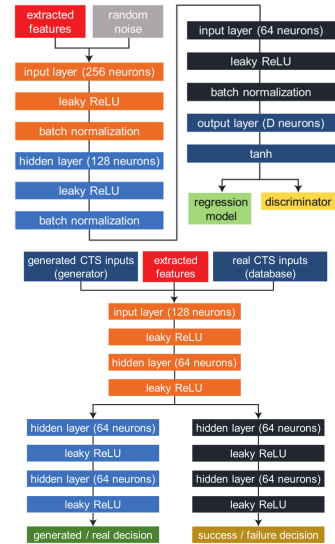
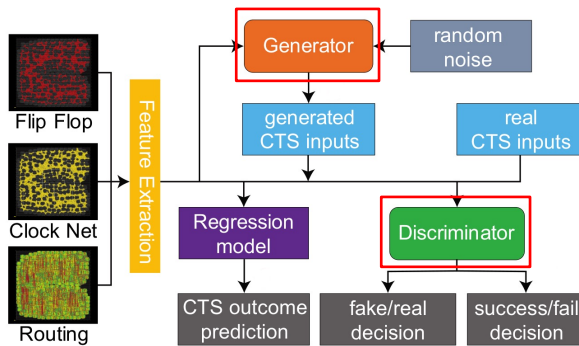
Y.-C. Lu, J. Lee, A. Agnesina, K. Samadi, and S. K. Lim, "GAN-CTS: A generative adversarial framework for clock tree prediction and optimization," *Proceedings of the IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 1–8, Nov 2019

112

GAN-CTS Framework: Generator/Discriminator

113

- Use Generative Adversarial Network to predict optimal CTS parameters
 - Generator network proposes candidate CTS parameters
 - Discriminator network tunes for the best candidate



Y.-C. Lu, J. Lee, A. Agnesina, K. Samadi, and S. K. Lim, "GAN-CTS: A generative adversarial framework for clock tree prediction and optimization," *Proceedings of the IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 1–8, Nov 2019

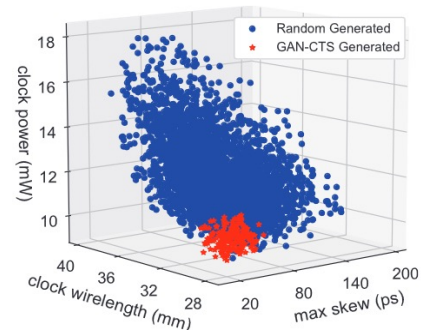
113

GAN-CTS Results

114

- Clock trees generated by GAN-CTS have more optimal clock power, skew, and wirelength as compared to the commercial tool
 - GAN-CTS provide improved metric scores compared to the auto-generated clock trees

netlist	CTS Metrics	Auto-Setting	GAN-CTS
aes	# inserted buffers	695	83 (-88.1%)
	clock power (mW)	20.34	9.86 (-51.5%)
	clock wirelength (mm)	34.09	27.78 (-18.5%)
	maximum skew (ns)	0.019	0.018 (-5.3 %)
nova	# inserted buffers	2617	524 (-79.9%)
	clock power (mW)	43.59	19.86 (-54.4%)
	clock wirelength (mm)	118.24	97.92 (-17.2%)
	maximum skew (ns)	0.031	0.033 (+6.4 %)



Y.-C. Lu, J. Lee, A. Agnesina, K. Samadi, and S. K. Lim, "GAN-CTS: A generative adversarial framework for clock tree prediction and optimization," *Proceedings of the IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 1–8, Nov 2019

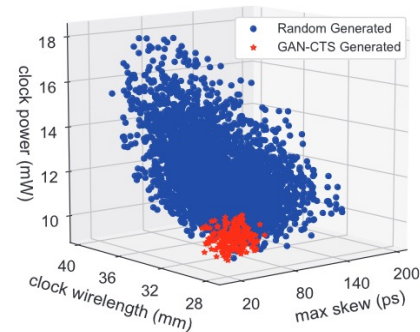
114

GAN-CTS Results

115

- Clock trees generated by GAN-CTS have more optimal clock power, skew, and wirelength as compared to the commercial tool
 - **GAN-CTS provide improved metric scores compared to the auto-generated clock trees**

netlist	CTS Metrics	Auto-Setting	GAN-CTS
aes	# inserted buffers	695	83 (-88.1%)
	clock power (mW)	20.34	9.86 (-51.5%)
	clock wirelength (mm)	34.09	27.78 (-18.5%)
	maximum skew (ns)	0.019	0.018 (-5.3 %)
nova	# inserted buffers	2617	524 (-79.9%)
	clock power (mW)	43.59	19.86 (-54.4%)
	clock wirelength (mm)	118.24	97.92 (-17.2%)
	maximum skew (ns)	0.031	0.033 (+6.4 %)



Y.-C. Lu, J. Lee, A. Agnesina, K. Samadi, and S. K. Lim, "GAN-CTS: A generative adversarial framework for clock tree prediction and optimization," *Proceedings of the IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 1–8, Nov 2019

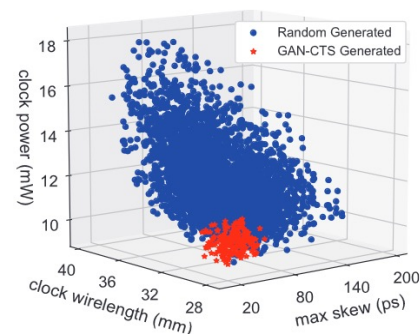
115

GAN-CTS Results

116

- Clock trees generated by GAN-CTS have more optimal clock power, skew, and wirelength as compared to the commercial tool
 - GAN-CTS provide improved metric scores compared to the auto-generated clock trees
- Confusion matrix of the successful and failed classification of the NOVA benchmark circuit by Discriminator D
 - Failure indicated a lower CTS metric score than the auto-setting generated clock tree
 - Accuracy: 0.947
 - F1-score: 0.952

netlist	CTS Metrics	Auto-Setting	GAN-CTS
aes	# inserted buffers	695	83 (-88.1%)
	clock power (mW)	20.34	9.86 (-51.5%)
	clock wirelength (mm)	34.09	27.78 (-18.5%)
	maximum skew (ns)	0.019	0.018 (-5.3 %)
nova	# inserted buffers	2617	524 (-79.9%)
	clock power (mW)	43.59	19.86 (-54.4%)
	clock wirelength (mm)	118.24	97.92 (-17.2%)
	maximum skew (ns)	0.031	0.033 (+6.4 %)



		Predictions		Total
		Success	Failure	
Ground Truths	Success	1848	111	1959
	Failure	74	1453	1527
	Total	1922	1564	3486



Y.-C. Lu, J. Lee, A. Agnesina, K. Samadi, and S. K. Lim, "GAN-CTS: A generative adversarial framework for clock tree prediction and optimization," *Proceedings of the IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 1–8, Nov 2019

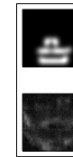
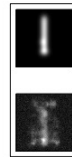
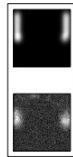
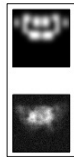
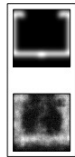
116

ML-guided Analog Routing Approach: GeniusRoute

117

- Algorithm: Variational auto-encoders (VAEs) trained on layout images
 - Encoder: map input image to low-dimensional space
 - Decoder: generate routing guidance
 - Label: routing region of nets
- Routing prediction: for a given placement, VAE predicts the probability map that a wire is placed in a region
- Routing algorithm: A* search algorithm guided by the trained VAE model
- Limitation: GeniusRoute is trained on a dataset consisting of comparators and amplifiers without generalizing to other analog circuit types

ground-truth



inference



K. Zhu, et al., "GeniusRoute: A New Analog Routing Paradigm Using Generative Neural Network Guidance," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp.1-8, Nov 2019

117

Case Studies

118

Case Study 1: Prediction of Downstream Circuit Metrics in Physical Design

- Timing Prediction, Interconnect Parasitic Prediction, Power Prediction

Case Study 2: Generation and Optimization of Circuits

- Automated Placement, Clock Network Synthesis, Routing

Case Study 3: Transfer learning approaches

Case Study 4: Large Language Models in Physical Design



118

Case Study 3: Transfer Learning for IC Design

119

- Transfer Learning involves applying knowledge gained from solving one problem and applying it to a different but related problem
- Key in accelerating the design process by leveraging pre-trained models
- Key Components
 - Source Task: The original task where the model is trained
 - Target Task: The new task where the model is applied
 - Knowledge Transfer: The process of adapting the model from the source task to the target task
- Benefits of Transfer Learning
 - Efficiency: Reduces computational resources and training time needed
 - Performance: Enhances model performance especially when data on the target task is limited
 - Versatility: Enables cross domain applications



119

Applications of Transfer Learning to IC Design

120

Cross Technology/ PDK Prediction

Utilizing models trained on older technology node to predict outcomes on newer technology node. This helps in adapting to new manufacturing processes faster.

Cross Tool Prediction

Applying knowledge learned from one EDA tool to enhance or predict the performance of another tool.

Cross Design Prediction

Using models trained on specific circuit designs (like memories or analog circuits) to predict characteristics of other types of designs (like logic or mixed-signal circuits).

Cross Corner Prediction

Adapting models that are trained for specific operating conditions (temperature and voltage corners) to predict performance across different conditions.

Cross Functionality Prediction

Employing data from one functional area (like timing analysis) to improve predictions in another (like power analysis).

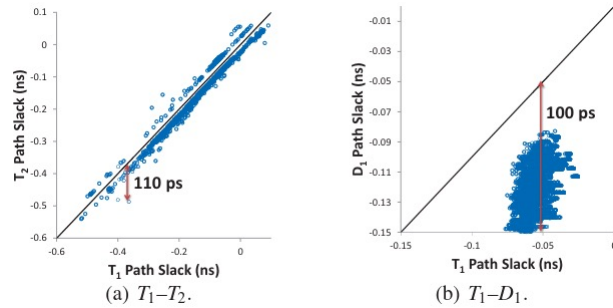


120

Deep Learning Methodology to Model Golden Signoff Timing

121

- Develop a machine learning-based tool, Golden Timer eXtension (GTX), to correct discrepancies in various timing metrics between different signoff timing analysis tools
- Correlate path slack between given two signoff timing tools (T1 and T2) and a design tool (D1)
 - Ensure consistency across tools
 - Cross-tool validation
 - Facilitate multi-vendor environments
 - Adapt to advanced technology nodes
 - Streamline collaboration and handoffs
- Precursor to model re-usability (transfer learning) across tools



S. S. Han, A. B. Kahng, S. Nath, & A. S. Vydyanathan, "A deep learning methodology to proliferate golden signoff timing," *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, March 2014

121

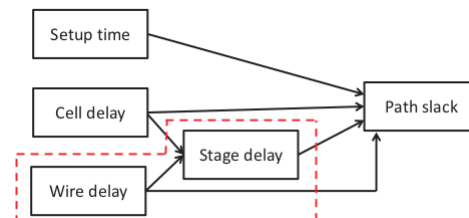
Deep Learning Methodology to Model Golden Signoff Timing

122

- Dataset
 - PDK: 28nm FDSOI and 45nm GS foundry libraries
- Model Architecture
 - Metric to Predict: Flip-flop setup time, cell arc delay, wire delay, stage delay, and path slack at timing endpoints
 - Hierarchical models using Random Forest and SVM
 - Model is "deep" because of the hierarchical approach

Testcase	# Instances (clock period in ns)	
	45nm	28nm
<i>aes_cipher_top</i>	18818 (1.0)	16688 (0.8)
<i>wb_dma_top</i>	3641 (0.5)	2349 (0.5)
<i>jpeg_encoder</i>	46702 (1.25)	53641 (0.67)
<i>leon3mp</i>	—	750854 (1.2)

- Feature set (numeric)
 - Capacitance (load, coupling, wire to ground)
 - Wire resistance
 - Cell slew (input, output)
 - Delay (cell, wire, stage)
 - Flip-flop setup time
 - Path slack



S. S. Han, A. B. Kahng, S. Nath, & A. S. Vydyanathan, "A deep learning methodology to proliferate golden signoff timing," *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, March 2014

122

Deep Learning Methodology to Model Golden Signoff Timing

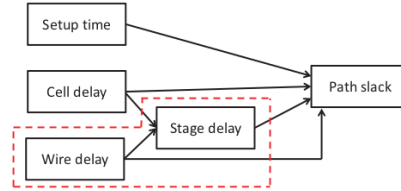
123

• Dataset

- PDK: 28nm FDSOI and 45nm GS foundry libraries

Testcase	# Instances (clock period in ns)	
	45nm	28nm
<i>aes_cipher_top</i>	18818 (1.0)	16688 (0.8)
<i>wb_dma_top</i>	3641 (0.5)	2349 (0.5)
<i>jpeg_encoder</i>	46702 (1.25)	53641 (0.67)
<i>leon3mp</i>	—	750854 (1.2)

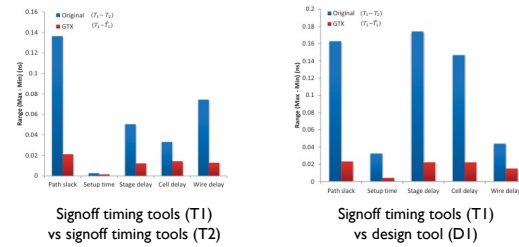
• Model Architecture



• Feature set (numeric)

- Capacitance (load, coupling, wire to ground)
- Wire resistance
- Cell slew (input, output)
- Delay (cell, wire, stage)
- Flip-flop setup time
- Path slack

• Results



S. S. Han, A. B. Kahng, S. Nath, & A. S. Vydyanathan, "A deep learning methodology to proliferate golden signoff timing," *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, March 2014

123

Transfer Learning with Domain Adaptation

124

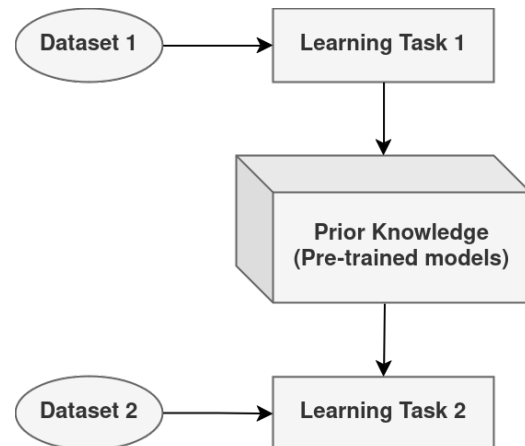
- Freeze a variable number of layers of the prior model(s)
- Retrain with a smaller dataset in the target domain/node

Algorithm 1 Transfer Training with Frozen Layers.

```

1: procedure TRAIN
2:    $U_t \leftarrow$  training data from target domain
3:    $P \leftarrow$  pre-trained network
4:    $l \leftarrow$  number of intermediate layers
5:    $h \leftarrow$  number of frozen layers
6:   freeze first  $h$  intermediate layers of  $P$ 
7:   for  $i \in \{h+1, h+2, \dots, l\}$ 
8:     retrain layer  $i$  of  $P$  on  $U_t$ 
9:   end for
10:  repeat until the maximum number of epochs reached
11: end procedure

```



124

Transfer Learning for Arrival Time Prediction

125

- Transfer setup
 - Source: 65 nm model
 - Target: 28 nm transfer model
 - Baseline: 28 nm model
- Dataset
 - Circuits: Six IWLS'05 benchmark circuits
 - PDK: Commercial 28 nm and 65 nm
 - Toolset: Synopsys DC and ICC2 Compiler
- Modeling and Evaluation Scenarios
 - Preliminary Scenario: Initial 65 nm model
 - Generate data for 65 nm
 - Train model for 65 nm
 - Scenario I: 28 nm model
 - Generate data for 28 nm
 - Train model for 28 nm
 - Scenario II: Predict 28 nm arrival times using the 65 nm model
 - Infer on 28 nm
 - No new dataset generation and training for 28 nm
 - Scenario III: Transfer model (65 nm to 28 nm)
 - Generate data for 28 nm
 - Fine tune model using 28 nm data

Design	No. of inputs	No. of outputs	No. of Gates		
			Sequential	Combinational	Total
ac97_ctrl	84	48	2199	9656	11855
aes_core	259	129	530	20265	20795
wb_conmax	1130	1416	770	28264	29034
des3_perf	234	64	8808	89533	98341
usb_funct	128	121	1746	11062	12808
pci	162	207	3359	13457	16816



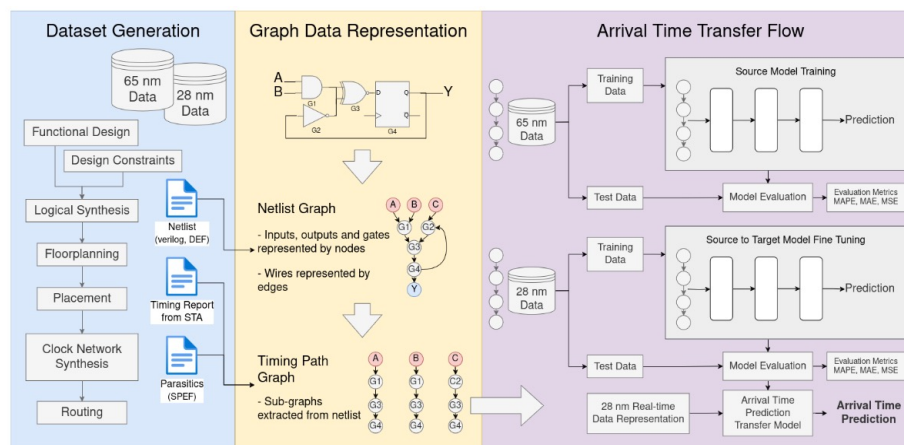
P. Shrestha and I. Savidis, "Transfer Learning of Arrival Time Prediction Models from a 65 nm to a 28 nm Process Node," *Proceedings of the IEEE Dallas Circuits and Systems Conference (DCAS)*, pp. 1–6, April 2024

125

Transfer Learning Framework

126

- Re-use of Dataset Generation and Graph Representation



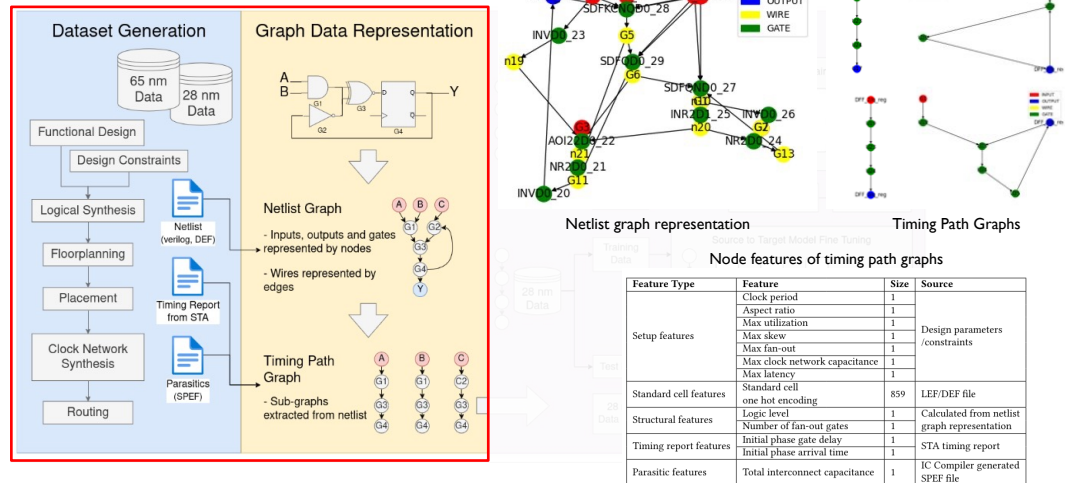
P. Shrestha and I. Savidis, "Transfer Learning of Arrival Time Prediction Models from a 65 nm to a 28 nm Process Node," *Proceedings of the IEEE Dallas Circuits and Systems Conference (DCAS)*, pp. 1–6, April 2024

126

Transfer Learning Framework

127

- Re-use of Dataset Generation and Graph Representation



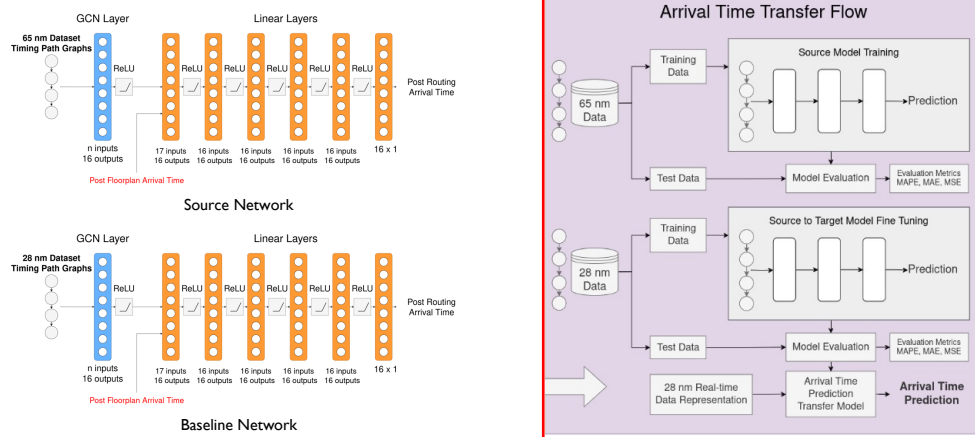
P. Shrestha and I. Savidis, "Transfer Learning of Arrival Time Prediction Models from a 65 nm to a 28 nm Process Node," *Proceedings of the IEEE Dallas Circuits and Systems Conference (DCAS)*, pp. 1–6, April 2024

127

Transfer Learning Framework

128

- Re-use of Dataset Generation and Graph Representation
- Identical source and baseline models



P. Shrestha and I. Savidis, "Transfer Learning of Arrival Time Prediction Models from a 65 nm to a 28 nm Process Node," *Proceedings of the IEEE Dallas Circuits and Systems Conference (DCAS)*, pp. 1–6, April 2024

128

Transfer Learning Framework

129

- Re-use of Dataset Generation and Graph Representation
- Identical source and baseline models
- Transfer tuned model
 - Freeze all layers for 65 nm model
 - Add transfer layer for 28 nm to fine tune

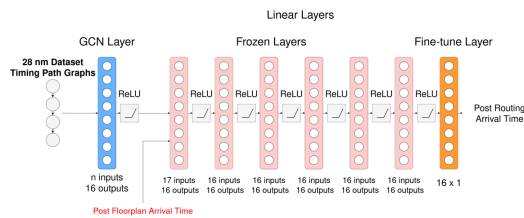
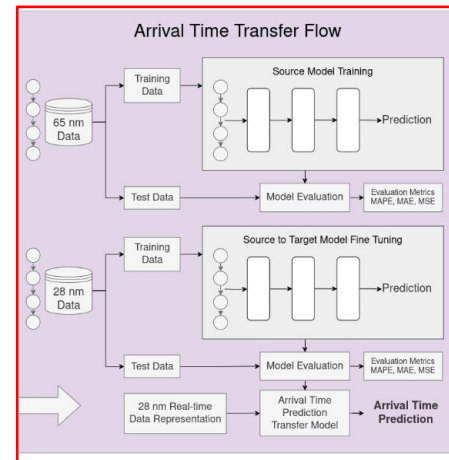


Fig: Transferred Network

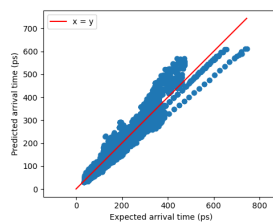


P. Shrestha and I. Savidis, "Transfer Learning of Arrival Time Prediction Models from a 65 nm to a 28 nm Process Node," *Proceedings of the IEEE Dallas Circuits and Systems Conference (DCAS)*, pp. 1–6, April 2024

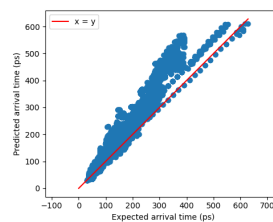
129

Transfer Learning Framework: Results

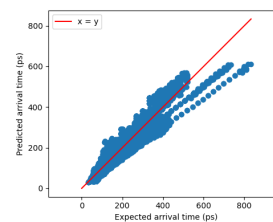
130



Expected vs Predicted Value:
Scenario I



Expected vs Predicted Value:
Scenario II



Expected vs Predicted Value:
Scenario III

	Scenario I (Baseline: Use of 28 nm)	Scenario II (Direct Use of 65 nm)	Scenario III (Transfer Model Fine-Tuning)
Model Performance	Lowest MAE, MAPE among scenarios	43.73% higher MAE, 42.69% higher MAPE compared to Scenario I	38.49% improvement in MAE, 28.16% improvement in MAPE compared to Scenario II
Time & Resources	Requires 4 hours and 23 minutes for data generation and training	No additional data generation or modeling required, time-efficient compared to Scenario I	Requires 1 hour and 25 minutes for dataset generation; 4 minutes for fine-tuning
Remarks	Best model performance Worst time and resource investment	Worst model performance No time and resource investment	Medium model performance Medium time and resource investment



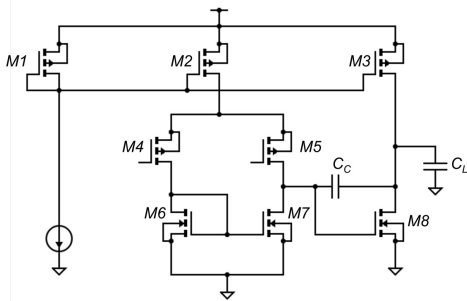
P. Shrestha and I. Savidis, "Transfer Learning of Arrival Time Prediction Models from a 65 nm to a 28 nm Process Node," *Proceedings of the IEEE Dallas Circuits and Systems Conference (DCAS)*, pp. 1–6, April 2024

130

Transfer Performance Modeling Across Technology Nodes for the Same Circuit

131

- Transfer learning is applied on models trained in 180 nm for the performance modeling of an op-amp in 65 nm
- Transfer learning significantly improves the sample efficiency for circuit performance modeling with simulation-based sizing data
 - Up to 50% improvement in MAE on test data



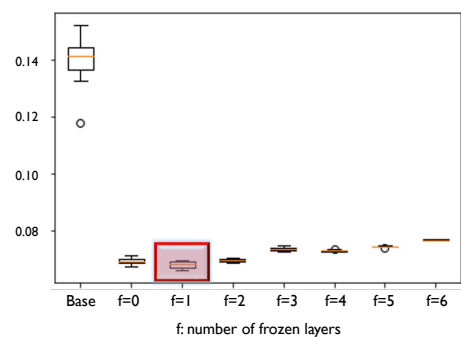
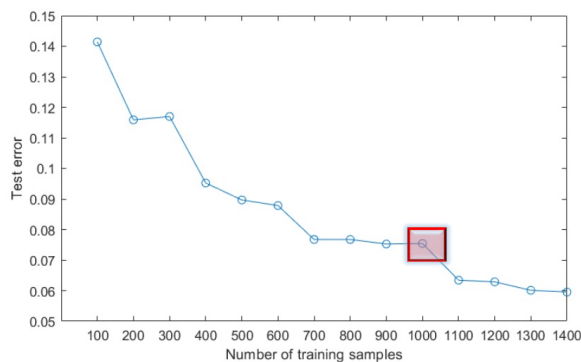
Z. Wu and I. Savidis, "Transfer Learning for Reuse of Analog Circuit Sizing Models Across Technology Nodes," *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, May 2022.

131

Comparison of Sample Efficiency for Training of the Gain Predictor

132

- Standalone training requires 1000 training samples to achieve test error of 0.076
- Transfer learning requires 100 samples to achieve test error of 0.07



Z. Wu and I. Savidis, "Transfer Learning for Reuse of Analog Circuit Sizing Models Across Technology Nodes," *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, May 2022.

132

Case Studies

133

Case Study 1: Prediction of Downstream Circuit Metrics in Physical Design

- Timing Prediction, Interconnect Parasitic Prediction, Power Prediction

Case Study 2: Generation and Optimization of Circuits

- Automated Placement, Clock Network Synthesis, Routing

Case Study 3: Transfer learning approaches

Case Study 4: Large Language Models in Physical Design



133

Natural Language as a Modality in EDA

134

- EDA has diverse natural language artifacts encoding both intent and flow state:
 - RTL + RTL comments + inline intent
 - Constraints and design assumptions
 - Engineering logs / tool diagnostics / reports (STA, P&R, DRC, IR, coverage)
 - Documentation, bug trackers, internal notes
 - Academic + industry publications / benchmarks
- These natural language sources are being used today for:
 - chatbots / design assistants (*ChipNeMo*, *ChatEDA*, *OpenROAD-Assistant*)
 - code + constraint generation (*ChipGPT*, *BetterV*, *RTLLM*, *VeriGen*)
 - error explanation + debug triage (*LLM-aided synthesis error explanation*)
 - spec extraction + semantic grounding (*SpecLLM*, *AssertLLM*)



134

VerilogEval: Evaluating Large Language Models for Verilog Code Generation

135

- Open benchmark for measuring functional correctness of LLM based Verilog generation
- Curated benchmarks
 - Benchmarks sourced from [HDLBits](#) (widely used digital design learning platform)
 - 156 Verilog tasks selected with clear specifications, known reference implementations, and deterministic simulation behavior
- Evaluation Process
 - LLMs are prompted with each task to generate Verilog
 - Generated Verilog is compiled + simulated under the same test harness
 - Correctness determined by functional simulation match
 - **Pass@k metric:** percentage of tasks where at least one of the top-k generated candidates simulates correctly

System Prompt (Optional):

You only complete chats with syntax correct Verilog code. End the Verilog module code completion with 'endmodule'. Do not include module, input and output definitions.

Question Prompt:

Implement the Verilog module based on the following description. Assume that signals are positive clock/clk edge triggered unless otherwise stated.

Problem Description:

Given an 8-bit input vector [7:0], reverse its bit ordering.

```
module top_module (
    input [7:0] in,
    output [7:0] out
);
```

Canonical Solution:

```
assign {out[0],out[1],out[2],out[3],out[4],
        out[5],out[6],out[7]} = in;
endmodule
```



L. Mingjie, N. Pinckney, B. Khailany, and H. Ren, "VerilogEval: Evaluating large language models for verilog code generation," *Proceedings of the IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 1–8, Oct 2023

135

VerilogEval: Evaluating Large Language Models for Verilog Code Generation

136

- Models compared
 - GPT-4, GPT-3.5, verilog-sft
- verilog-sft
 - Take codegen-16B (open code LLM) and supervised fine tune it on Verilog prompt → ground truth Verilog pairs (HDLBits style)
 - Teaches the model the correct mapping from spec → implementation in the HDL domain
- Evaluation Splits
 - VerilogEval-machine: canonical HDLBits prompts
 - VerilogEval-human: human-rephrased natural language versions of the same tasks

Model	VerilogEval-machine			VerilogEval-human		
	pass@1	pass@5	pass@10	pass@1	pass@5	pass@10
gpt-3.5	46.7	69.1	74.1	26.7	45.8	51.7
gpt-4	60.0	70.6	73.5	43.5	55.8	58.9
verilog-sft	46.2	67.3	73.7	28.8	45.9	52.3



L. Mingjie, N. Pinckney, B. Khailany, and H. Ren, "VerilogEval: Evaluating large language models for verilog code generation," *Proceedings of the IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 1–8, Oct 2023

136

ChipNeMo: Domain-Adapted LLMs for Chip Design

137

- NVIDIA's domain adapted large language model

- **Base LLM Model:**

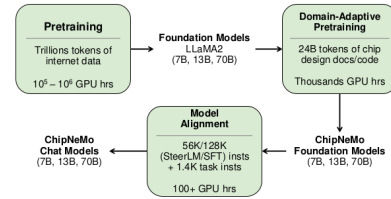
- start with huge generic internet scale pretraining → Llama2 foundation models (7B / 13B / 70B)

- **Domain-Adaptive Pretraining (DAPT):**

- pretrain based LLM on domain-specific corpora (EDA code, logs, specs) so it becomes fluent on the technical domain

- **Model Alignment:**

- supervised fine-tune on domain instruction + task data so the model behaves like a chip design engineer



- DAPT = what the model knows (domain fluency)
- Alignment = how the model responds (domain correct behavior)

Data Source Type	Data Percentage (%)	Data Tokens (B)	Training Percentage (%)	Training Tokens (B)
Bug Summary	9.5%	2.4	10.0%	2.4
Design Source	47.0%	11.9	24.5%	5.9
Documentation	17.8%	4.5	34.0%	8.2
Verification	9.1%	2.3	10.4%	2.5
Other	7.9%	2.0	12.0%	2.9
Wikipedia	5.9%	1.5	6.2%	1.5
Github	2.8%	0.7	3.0%	0.7
Total	100.0%	25.3	100.0%	24.1



L. Mingjie, T.-D. Ene, R. Kirby, C. Cheng, N. Pinckney, R. Liang, J. Alben et al., "ChipNeMo: Domain-Adapted LLMs for Chip Design", preprint arXiv:2311.00176, pp. 1–23, Apr 2024

137

ChipNeMo Use cases

138

- Engineering Assistant Chatbot

- help designers reason about specs, flows, constraints, error conditions, and next-step actions in natural language

- EDA Tool Script Generation

- generate/modify tool scripts (Tcl, synthesis, P&R recipes, timing queries, etc.) for specific flows and tool semantics

- Bug Summarization & Analysis

- read logs, reports, regressions, failures
 - produce summaries, diagnoses root-cause, and proposes next corrective actions



L. Mingjie, T.-D. Ene, R. Kirby, C. Cheng, N. Pinckney, R. Liang, J. Alben et al., "ChipNeMo: Domain-Adapted LLMs for Chip Design", preprint arXiv:2311.00176, pp. 1–23, Apr 2024

138

ChipNeMo Use cases

139

- **Engineering Assistant Chatbot**
 - help designers reason about specs, flows, constraints, error conditions, and next-step actions in natural language
- EDA Tool Script Generation
 - generate/modify tool scripts (Tcl, synthesis, P&R recipes, timing queries, etc.) for specific flows and tool semantics
- Bug Summarization & Analysis
 - read logs, reports, regressions, failures
 - produce summaries, diagnoses root-cause, and proposes next corrective actions

Question:

why do we add retiming stages?

Answer:

Retiming stages are added to improve the timing of the design and to account for the physical distance between different parts of the chip. By adding retiming stages, we can reduce the amount of time it takes for signals to travel between different parts of the chip, which can help to improve the overall performance of the design. Additionally, retiming stages can also help to reduce the amount of noise and interference that can occur on the signals, which can further improve the performance of the design.



L. Mingjie, T.-D. Ene, R. Kirby, C. Cheng, N. Pinckney, R. Liang, J. Alben et al., "ChipNeMo: Domain-Adapted LLMs for Chip Design", preprint arXiv:2311.00176, pp. 1–23, Apr 2024

139

ChipNeMo Use cases

140

- Engineering Assistant Chatbot
 - help designers reason about specs, flows, constraints, error conditions, and next-step actions in natural language
- **EDA Tool Script Generation**
 - generate/modify tool scripts (Tcl, synthesis, P&R recipes, timing queries, etc.) for specific flows and tool semantics
- Bug Summarization & Analysis
 - read logs, reports, regressions, failures
 - produce summaries, diagnoses root-cause, and proposes next corrective actions

Question:

Write a *TOOL2* tcl proc to find intersecting logic from pins '\$pinA' to pin '\$pinB'.

Answer:

```
#Sure, here is the code:
proc myfunc {pinA pinB} {
    # Get all the pins in the fanout of the
    ↪ given pin
    set fanout_pins [fanout -from $pinA]
    # Get all the pins in the fanout of the
    ↪ given pin
    set fanin_pins [fanin -to $pinB]
    # Get the intersection of the two
    ↪ collections
    set intersection [common_logic
    ↪ $fanout_pins $fanin_pins]
    # Return the intersection
    return $intersection
}
```



L. Mingjie, T.-D. Ene, R. Kirby, C. Cheng, N. Pinckney, R. Liang, J. Alben et al., "ChipNeMo: Domain-Adapted LLMs for Chip Design", preprint arXiv:2311.00176, pp. 1–23, Apr 2024

140

ChipNeMo Use cases

141

- Engineering Assistant Chatbot
 - help designers reason about specs, flows, constraints, error conditions, and next-step actions in natural language
- EDA Tool Script Generation
 - generate/modify tool scripts (Tcl, synthesis, P&R recipes, timing queries, etc.) for specific flows and tool semantics
- Bug Summarization & Analysis
 - read logs, reports, regressions, failures
 - produce summaries, diagnoses root-cause, and proposes next corrective actions

Prompt:
 BugId : 002
 Synopsis : Request power/current measurement on fabrics using fabric test module
 Module : Silicon - DFT
 Description :
 Hi all,
 We want to evaluate silicon current (power) consumed by fabric macros in a chip. This will help us correlate this power to pre-Si power estimates.
 Here are the conditions we want to look at for this data:
 Fabric test module activity: 0/25/50/75/100%
 Corners:
 typical.low.voltage
 typical.high.voltage
 We want to isolate fabric current (power) as much as possible, so we want to enable only the logic needed for fabric test modules and the fabrics themselves.
 If this information can be broken down per module (there are 6 in the chip), that will help.
 Severity : 5-Performance
 Priority : 5-Normal
 Comments :
 #1 : Author : User A
 Is this a requirement only for current or for future chips as well ?
 #2 : Author : User B
 There will likely be interest in future chips to perform this exercise as we push the boundaries of fabric power density for better area.
 #3 : Author : User A
 What is needed in this bug ?
 @User B It is not clear.
 Can you list the specific needs with timelines and priority

Technical Summary:
 In the internal bug management system, a bug has been filed regarding the power and current measurements of the fabrics in the latest chip. The request is to evaluate the silicon current (power) consumed by the fabric macros in the latest chip and to correlate this power to pre-Si power estimates. The conditions for the measurements are the fabric test module activity at 0/25/50/75/100% and the corners are typical.low.voltage and typical.high.voltage. The data needs to be broken down per module (there are 6 in the latest chip). Only the logic needed for fabric test modules and the fabrics themselves need to be enabled.

Managerial Summary:
 User A asks User B if the need for evaluating the silicon current of fabric macros in a chip is only for that specific chip or for future ones as well. User B replies that there will likely be interest in future chips to perform this exercise as they push the boundaries of fabric power density for better area. User A then asks what is needed in this bug and asks for specific needs with timelines and priority.

Task Assignment:
 User B



L. Mingjie, T.-D. Ene, R. Kirby, C. Cheng, N. Pinckney, R. Liang, J. Alben et al., "ChipNeMo: Domain-Adapted LLMs for Chip Design", preprint arXiv:2311.00176, pp. 1–23, Apr 2024

141

ChipNeMo Benchmark Evaluation

142

Task	Benchmark	Evaluation Measure	Result Summary
Engineering Assistant Chatbot	88 engineering questions (Specs / Testbench / Build)	Likert scale: human raters score 1 (poor) → 7 (excellent)	<ul style="list-style-type: none"> • Moderate, clear improvement from domain-adaptive pretraining • Higher judged usefulness and correctness vs. base LLM
EDA Tool Script Generation	EDA tool scripting tasks across difficulty tiers	Pass@5 accuracy (success if any of top 5 answers is correct)	<ul style="list-style-type: none"> • Domain-adaptive pretraining delivered the largest lift • Final ChipNeMo achieved ~20–35% Pass@5 gains vs. base LLM • Gains held across script evaluation categories
Bug Summarization and Analysis	Bug logs / regression failures / engineering failure reports	Human scoring + comparison against GPT-4 scoring baseline	<ul style="list-style-type: none"> • ChipNeMo slightly surpassed GPT-4 on clarity and actionable debugging • Domain adaptation consistently improved quality on this task



L. Mingjie, T.-D. Ene, R. Kirby, C. Cheng, N. Pinckney, R. Liang, J. Alben et al., "ChipNeMo: Domain-Adapted LLMs for Chip Design", preprint arXiv:2311.00176, pp. 1–23, Apr 2024

142

CVDP: Comprehensive Verilog Design Problems Benchmark (2025)

143

- First comprehensive benchmark set for Verilog RTL design tasks, debugging, verification, assertions, and comprehension
- 783 human-authored problems across 13 categories enabling broad testing coverage
- Supports both non-agentic and agentic evaluation with dockerized agents

ID	Category Description	Volume	
		Non-Agentic	Agentic
Code Generation			
cid02	RTL - Code Completion	94	0
cid03	RTL - Natural Language Spec to Code	78	37
cid04	RTL - Code Modification	56	26
cid05	RTL - Spec to RTL (Module Reuse)	0	26
cid07	RTL - Code Improvement (Linting/QoR)	41	0
cid12	Design Verification - Testbench Stimulus Gen.	68	18
cid13	Design Verification - Testbench Checker Gen.	53	18
cid14	Design Verification - Assertion Generation	68	30
cid16	Design Verification - Debugging / Bug Fixing	36	11
Code Comprehension			
cid06	Correspondence - RTL to/from Specification	34	0
cid08	Correspondence - Testbench to/from Test Plan	29	0
cid09	Question & Answer - RTL	34	0
cid10	Question & Answer - Testbench	26	0
	Total # of Problems	617	166



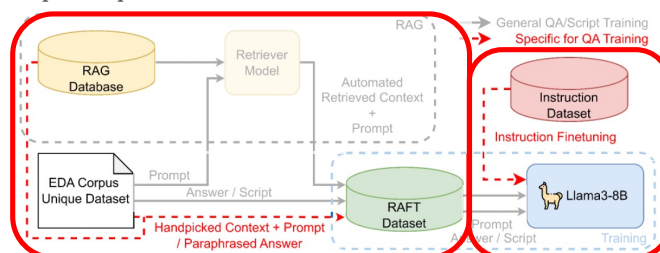
N. Pinckney, C. Deng, C.-T. Ho, Y.-D. Tsai, M. Liu, W. Zhou, B. Khailany, and H. Ren, "Comprehensive Verilog Design Problems: A Next-Generation Benchmark Dataset for Evaluating Large Language Models and Agents on RTL Design and Verification.", arXiv preprint arXiv:2506.14074, pp. 1–16, Jun 2025

143

OpenROAD-Assistant

144

- OpenROAD: A fully open-source physical design platform covering RTL-to-GDSII flow
- Objective: Fine tune a Retrieval Augmented Generation (RAG) model
 - Understands physical design intent in natural language
 - Generates correct, grounded OpenROAD physical design commands and scripts
- EDA Corpus used in RAG tuning
 - OpenROAD documentation, OpenROAD API reference, OpenROAD example scripts/flows
 - Hand curated PD examples + queries built around those APIs



U. Sharma, B.-Y. Wu, S. R. D. Kankipati, V. A. Chhabria, and A. Rovinski, "Openroad-assistant: An open-source large language model for physical design tasks.", In Proceedings of the ACM/IEEE International Symposium on Machine Learning for CAD (MLCAD), pp. 1–7, Sep 2024

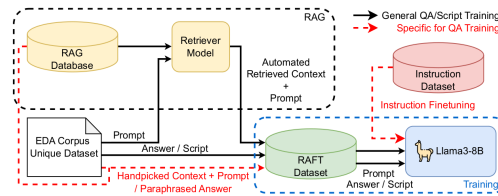
144

OpenROAD-Assistant – Model Tuning

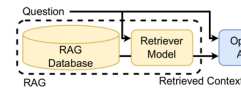
145

- Fine-tuning approach
 - base model: Llama3-8B
 - RAFT (Retrieval-Aware Fine Tuning) retrieval + SFT combined

- Retrieval-Aware Fine Tuning (RAFT)
 - retrieved OpenROAD docs/APIs/script examples are inserted into the model's input during training
 - model learns to ground generation on real OpenROAD semantics (not hallucinated commands / flags)



- Supervised Fine Tuning (SFT)
 - train on (retrieval context + NL prompt) → correct OpenROAD script / answer pairs
 - teaches exact mapping from natural language → valid physical design tool usage



- RAFT teaches where the truth comes from
- SFT teaches how to generate the correct answer using the truth.



U. Sharma, B.-Y. Wu, S. R. D. Kankipati, V. A. Chhabria, and A. Rovinski, "Openroad-assistant: An open-source large language model for physical design tasks.", In Proceedings of the ACM/IEEE International Symposium on Machine Learning for CAD (MLCAD), pp. 1–7, Sep 2024

145

Results

146

- Test Data: held-out OpenROAD PD tasks, scripts, and PD Q/A prompts derived from the OpenROAD EDA corpus.
 - Script Adapter → generates OpenROAD Tcl PD scripts
 - Q/A Adapter → answers physical design questions in natural language
- OpenROAD-Assistant is highest across both adapters:
 - Script adapter: 23 correct pass@1 / 24 correct pass@3 out of possible 30
 - Q/A adapter: highest BERTScore + BARTScore across Precision / Recall / F1 metrics
 - frontier models (Claude3 / GPT-4 Omni / Llama3) fail on script correctness

Category or Metrics		Script adapter						QA adapter					
		pass@1			pass@3			BERTScore			BARTScore		
		Correct	Partial	Wrong	Correct	Partial	Wrong	Precision	Recall	F1	Precision	Recall	F1
Found- ation models	Claude3	0	2	28	0	3	27	0.8169	0.8337	0.8251	0.4681	0.5091	0.4855
	GPT-4 Omni	1	1	28	4	3	23	0.8558	0.8549	0.8552	0.5634	0.5778	0.5689
	Llama3	0	0	30	0	0	30	0.8354	0.8454	0.8403	0.5222	0.5401	0.5301
Ablation studies	FT GPT-3.5 Turbo	6	6	18	11	4	15	0.8441	0.8441	0.8435	0.5514	0.5489	0.5393
	FT Llama3	13	4	13	14	3	13	0.8568	0.8463	0.8514	0.5899	0.5576	0.5714
	RAG FT Llama3	17	5	8	19	5	6	0.8690	0.8641	0.8655	0.6206	0.6573	0.7099
	RAFT Code Llama	20	3	7	20	3	7	-	-	-	-	-	-
OpenROAD-Assistant		23	6	1	24	5	1	0.9831	0.9855	0.9843	0.9517	0.9604	0.9557



U. Sharma, B.-Y. Wu, S. R. D. Kankipati, V. A. Chhabria, and A. Rovinski, "Openroad-assistant: An open-source large language model for physical design tasks.", In Proceedings of the ACM/IEEE International Symposium on Machine Learning for CAD (MLCAD), pp. 1–7, Sep 2024

146

Outline of Presentation

159

- Introduction to Electronic Design Automation
- Machine learning techniques
- Case studies
- **Standardizing ML for digital EDA**
- Conclusions

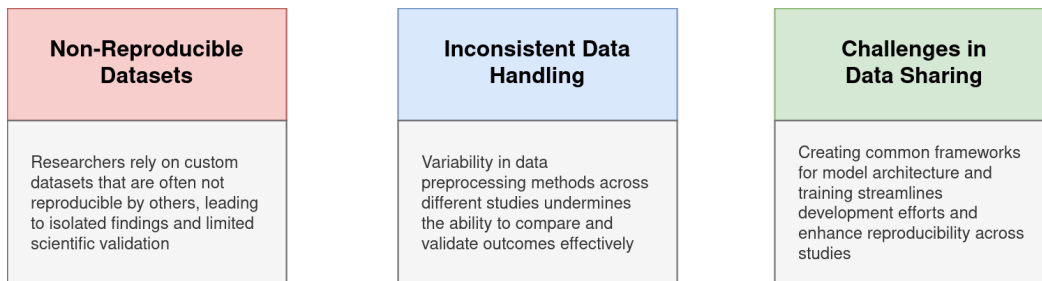


159

Challenges in ML for EDA: Lack of standardized dataset

160

- Absence of standardized, open datasets in physical design space is a significant barrier to the advancement and validation of machine learning models
- Key Issues



160

Digital Synthesis Toolflows: Commercial Digital Synthesis Tools

161

SYNOPSYS®

cadence®

Toolset	Synopsys	Cadence
Logical Synthesis	DC Compiler	Genus
Physical Design	IC Compiler, FusionCompiler	Innovus
Signoff	PrimeTime, StarRC, PrimePower, Formality ECO	Tempus, Joules RTL, Conformal Low Power, Litmus
RTL Simulation	VCS	Xcelium
Circuit Simulation	HSPICE, FineSim	Spectre
Custom Design	Custom Compiler	Virtuoso
Verification	Verdi	JasperGold
Power Analysis	PrimePower	Voltus
DFT and Test	DFTMAX, TetraMAX	Modus
Yield Optimization	Yield Explorer	Yield Enhancer
Analog/Mixed-Signal Design	CustomSim, Galaxy Custom Designer	Spectre, Virtuoso AMS Designer
Process and Device Simulation	Sentaurus	(No direct equivalent)



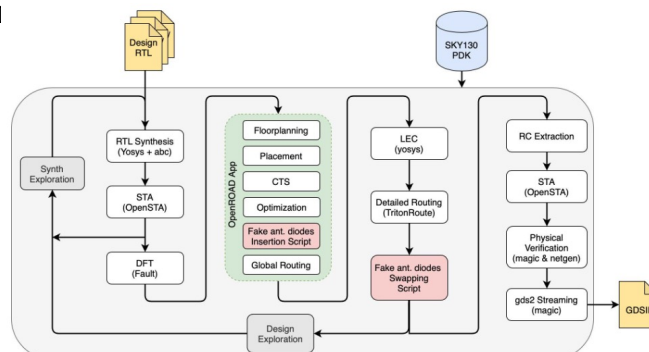
Source: <https://www.synopsys.com/implementation-and-signoff.html>
https://www.cadence.com/en_US/home/tools/digital-design-and-signoff.html

161

Digital Synthesis Toolflow: OpenLANE

162

- Open source, open access
- OpenROAD for RTL-GDSII flow
 - Rapid architecture and design space exploration
 - Early prediction of QoR
 - Detailed pl



DESIGN STEP	EDA Tool
Logic Synthesis	Yosys and ABC
DFT Scan Insertion	Fault
DFT ATPG	Fault
Formal Verification	Yosys
Placement	RePlace and OpenDP
Routing	FastRoute and TritonRoute
CTS	TritonCTS
Extraction	Magic
Timing Analysis	OpenSTA
Chip Floorplanning	PADGen
LVS	Netgen
DRC	Magic
GDS Streaming Out	Magic



M. Shalan and T. Edwards, "Building OpenLANE: A 130nm OpenROAD-based Tapeout-Proven Flow", *Proceedings of the IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 1–8, Nov 2020

162

Comparison of Digital Synthesis Platforms

163

	Open source/software?	PDK Support	Cost of use	3D Integration	Purpose	Silicon Proven?
Synopsys	No	High	Paid	Yes	Commerical	Yes
Cadence	No	High	Paid	Yes	Commerical	Yes
OpenLANE	Yes	Low	Free	No	Academic	Yes

- Open Source PDKs
 - Skywater 130 nm
 - IHP 130 nm
 - ASAP 7 nm
 - Global Foundries 180 nm
 - Nangate 45 nm
 - ICSprout 55 nm
- Closed PDKs
 - TSMC 16nm/28nm/65nm
 - Global Foundries 22nm/130nm
 - Intel 16nm



163

Standardized Training and Evaluation Protocols in ML for EDA

164

- Opportunities for Improvement

Establishing Ground Truths and Baselines	Harmonizing Train/Test Split Methodologies	Unifying Model Architectures and Training Processes	Evaluation Metrics and Procedures
Developing industry-wide accepted ground truths and baselines leads to more robust model comparisons and validations, accelerating the adoption of ML in EDA	Standardizing how data is divided for training and testing improves model fairness and performance consistency across diverse applications	Creating common frameworks for model architecture and training streamlines development efforts and enhance reproducibility across studies	Defining clear, universally accepted metrics and evaluation procedures simplifies the assessment of ML models, promoting transparency and reliability

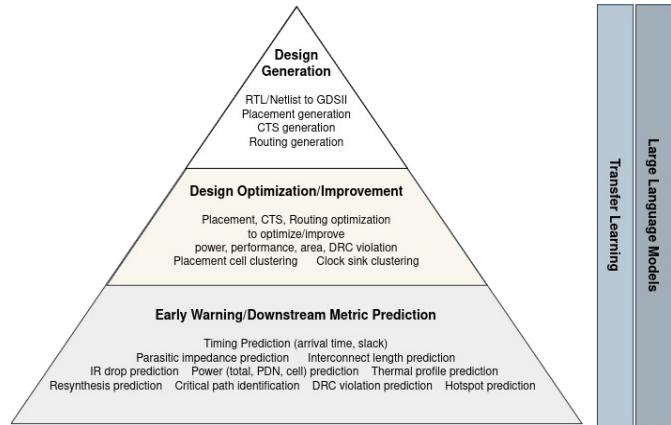


164

End-to-End Machine Learning for EDA

165

- Dataset modeling and benchmarking
 - Standardized dataset and data models
 - Metric of coverage, generalization
 - Benchmarks on common problems and models
- ML modelling and training framework
 - Feature engineering pipeline
 - Modeling architecture (GNNs, autoencoders, multi-modal, etc.)
 - Evaluation scheme
 - Hyper parameter tuning
 - Feature importance
- Application space
 - Leaderboard



- Vision/hope: open repository with production ready designs that are encrypted and secured but allow for ML design research by the community

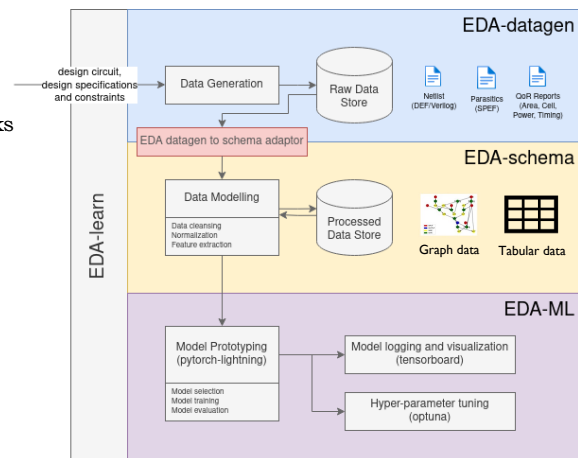


165

EDA-learn

166

- Data generation, representation, ML framework
- EDA-datasetgen
 - Physical design flow automation for parameterized large scale dataset generation
 - Run EDA tools to generate designs with public benchmarks
 - Toolboxes:
 - Parsers for standard formats of design files
 - Python interfaces
- EDA-schema
 - Property graph data-model schema for circuit data representation
- EDA-ML
 - Rapid prototyping and evaluation for EDA based machine learning models

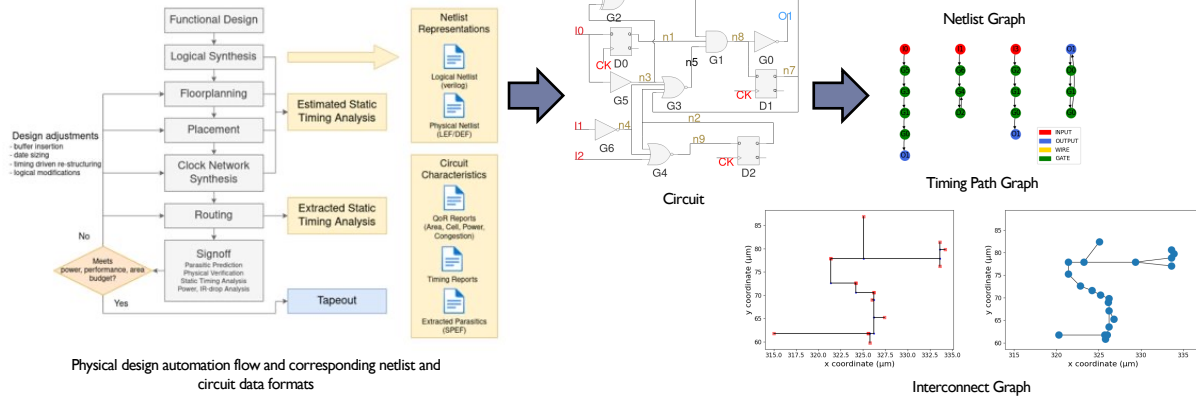


166

EDA-schema

167

- Standardized format of representing circuits
- Property graph data-model schema incorporates circuit information
 - Structural data of the circuit
 - Performance metrics from reports



P. Shrestha, A. Aversa, S. Phatharodom, and I. Savidis, "EDA-schema: A graph datamodel schema and open dataset for digital design automation," *Proceedings of the ACM Great Lakes Symposium on VLSI (GLSVLSI)*, pp. 1–8, June 2024

167

EDA-schema: Entity Relationship Diagram

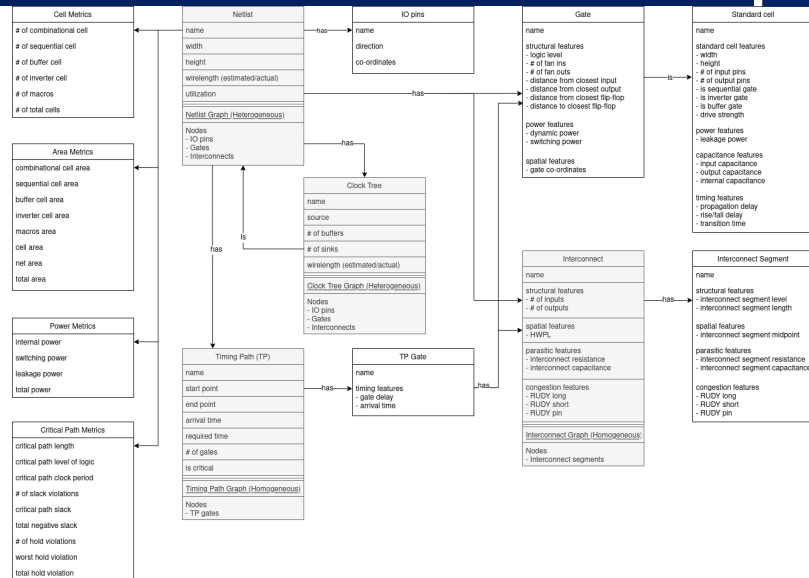
168

- Key components

- Netlist
- Clock Tree
- Timing Path
- Interconnect

- Features after each design phase

- Floorplan
- Placement
- CTS
- Routing



P. Shrestha, A. Aversa, S. Phatharodom, and I. Savidis, "EDA-schema: A graph datamodel schema and open dataset for digital design automation," *Proceedings of the ACM Great Lakes Symposium on VLSI (GLSVLSI)*, pp. 1–8, June 2024

168

EDA-schema: Feature Analysis

169

- Features used for machine learning driven circuit prediction problems in prior work.

Objective/Paper	Available Feature Set
placement parameter optimization [R1]	no. of cells, no. of nets, no. of IOs, <u>no. of nets with fanout ≤ 10</u> , no. of flip-flops, total cell area, no. of macros, macro area
pre-placement net length estimation [R2]	net's driver's area, fan-in and fan-out size
total design power prediction [R3]	minimum /maximum slack, worst input/output transition, switching power of driving net, cell switching power, cell internal power, cell leakage power
pre-routing timing prediction [R4]	driver and sink capacitance, sink locations, <u>distance between driver and the target sink</u> , max driver input transition
arrival time prediction [R5]	standard cell functionality, logic level, no. of fan-out, gate delay, interconnect capacitance, arrival time
interconnect parasitic prediction [R6]	interconnect capacitance, interconnect length, interconnect position, <u>pin density, net density</u>
wire parasitics and timing prediction [R7]	half perimeter wire length (HPWL), no. of sinks, congestion estimates, rise transition, fall transitions, interconnect length, interconnect RC

Reference:

- [R1] A. Agnesina, K. Chang, and S. K. Lim, "VLSI placement parameter optimization using deep reinforcement learning," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–9, Nov. 2020.
- [R2] Z. Xie, R. Liang, X. Xu, J. Hu, Y. Duan, and Y. Chen, "Net2: A graph attention network method customized for pre-placement net length estimation," *Proceedings of the Asia and South Pacific Design Automation Conference (ASPDAC)*, pp. 671–677, Jan. 2021.
- [R3] Y.-C. Lu, W.-T. Chan, V. Khandelwal, and S. K. Lim, "Driving early physical synthesis exploration through end-of-flow total power prediction," *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, pp. 97–102, Sept. 2022.
- [R4] E. C. Barboza, N. Shukla, Y. Chen, and J. Hu, "Machine learning-based pre-routing timing prediction with reduced pessimism," *Proceedings of the Design Automation Conference (DAC)*, pp. 1–6, Jun. 2019.
- [R5] P. Shrestha, S. Phatharodom, and I. Savidis, "Graph representation learning for gate arrival time prediction," *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, pp. 127–133, Sept. 2022.
- [R6] P. Shrestha and I. Savidis, "Graph representation learning for parasitic impedance prediction of the interconnect," *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, May 2023.
- [R7] V. A. Chhabra, W. Jiang, A. B. Kahng, and S. S. Saponekar, "From global route to detailed route: ML for fast and accurate wire parasitics and timing prediction," *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, pp. 7–14, Sept. 2022.



169

Standardized Dataset

170

- Open Dataset

- Designs: 20 IWLS'05 benchmark circuits
- PDK: Skywater 130nm
- Design Toolset: OpenROAD
- Setup parameters

Dataset constraints and parameters

Parameters	Values or ranges	# of Samples
Clock periods (ns)	{0.5, 1, 2, 5}	4
Aspect ratio	{0.5, 0.75, 1}	3
Max utilization	{0.3, 0.4, 0.5}	3
Max skew (ns)	{0.01 - 0.2}	1 random sample
Max fanout	{50 - 250}	1 random sample
Max clock network capacitance (pF)	{0.05 - 0.3}	1 random sample
Max latency (ns)	{0 - 1}	1 random sample
Total circuits per design		36
Overall circuits in dataset		36 * 20 = 720
No. of gates in dataset		7,468,228
No. of nets in dataset		7,726,920
No. of timing paths in dataset		1,561,795

IWLS'05 benchmark circuit characteristics

Circuits	No. of inputs	No. of outputs	Median no. of gates	No. of flip-flops	Median no. of nets	Median no. of timing paths	Time of Computation					Memory		
							Logical Synthesis	Floorplan	Placement	CTS	Routing	Total	Raw data	Processed data
ae97_ctrl	84	48	13124	2211	6153	2827	11m	7m	23m	8m	8m	58m	12G	1.435G
aes_core	259	129	20884	550	13613	1195	33m	8m	27m	8m	8m	1h 25m	11G	1.166G
des3_area	240	64	3977	64	2347	116	5m	5m	17m	5m	5m	40m	2.9G	0.298G
des3_perf	234	64	100455	8808	56818	16287	1h 39m	20m	1h 53m	27m	25m	4h 46m	85G	8.221G
ethernet	96	115	69541	10544	35649	20220	35m	16m	1h 18m	29m	28m	3h 5m	87G	8.907G
fpu	70	40	23228	643	14370	1669	2h 59m	9m	33m	9m	8m	4h 0m	14G	9.215G
ide	19	14	1053	129	485	196	5m	5m	15m	5m	5m	35m	1.9G	0.054G
mem_ctrl	115	152	9431	1051	5081	3422	13m	6m	21m	6m	6m	55m	7.6G	1.676G
pci	162	207	21612	3220	10769	3770	15m	8m	33m	8m	8m	1h 14m	15G	1.543G
sasc	16	12	792	118	340	180	2m	5m	15m	5m	5m	34m	1.8G	0.040G
simple_spi	16	12	1009	131	473	5311	2m	5m	15m	5m	5m	35m	1.9G	1.337G
spi	47	45	3091	229	1631	493	4m	5m	16m	5m	5m	38m	3G	0.187G
ss_pcm	19	9	640	87	294	109	2m	5m	16m	5m	5m	34m	1.6G	0.029G
systemaes	260	129	9565	670	5630	1436	6m	19m	6m	6m	6m	47m	7G	0.702G
systemcdes	132	65	2722	190	1343	446	4m	5m	16m	5m	5m	37m	2.6G	0.150G
tv80	14	32	7499	364	4620	3008	17m	6m	19m	6m	6m	55m	3.1G	1.052G
usb_func1	128	121	14792	1740	7963	159	11m	6m	21m	6m	6m	53m	6.1G	0.487G
usb_phy	15	18	674	108	275	141	2m	5m	15m	5m	5m	34m	1.7G	0.033G
vga_lcd	89	109	106047	17059	52511	42493	2h 26m	24m	2h 29m	1h 3m	1h 2m	7h 25m	160G	17.744G
vb_dma	217	215	4241	521	1996	1136	21m	6m	16m	5m	6m	56m	11G	0.328G
total							10h 20m	2h 52m	1h 28m	3h 49m	3h 47m	32h 20m	428G	82.836G

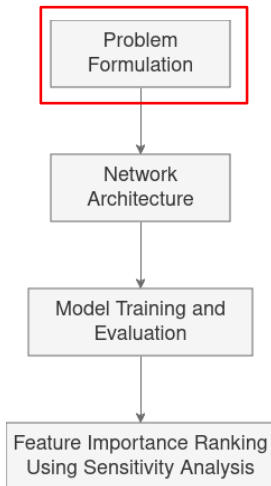


P. Shrestha, A. Aversa, S. Phatharodom, and I. Savidis, "EDA-schema: A graph datamodel schema and open dataset for digital design automation," *Proceedings of the ACM Great Lakes Symposium on VLSI (GLSVLSI)*, pp. 1–8, June 2024

170

Generalized ML Flow – Problem Formulation

171

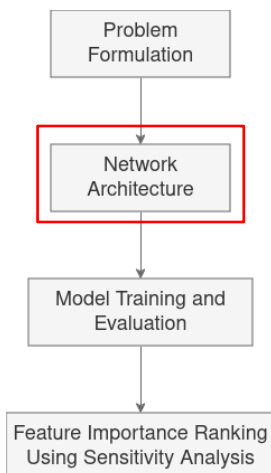


- Formulation of the problem includes:
 - Metric to be predicted
 - Initial phase of the prediction
 - Post-floorplan or post-placement
 - Final phase of the prediction
 - Post-routing
 - Appropriate graph representation
 - Whether the prediction is performed at graph level or node level
 - Baseline metric
 - Serves as a reference for the evaluation of the effectiveness of the model

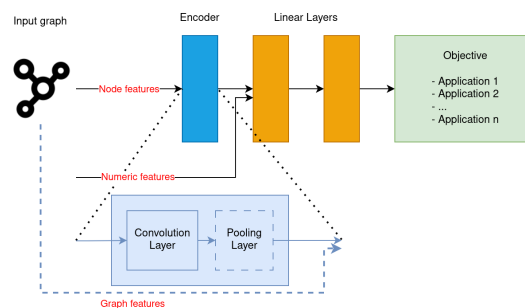
171

Generalized ML Flow – Network Architecture

172



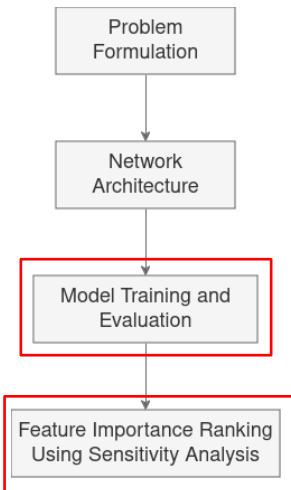
- Generalized template of the network architecture



172

Generalized ML Flow – Model Evaluation & Feature Importance Ranking

173



- Feature importance ranking
 - Evaluate importance of a feature on a model by leaving one feature out of each training run
 - Primary features are identified that significantly influence the accuracy of the model
 - Removing ineffective or redundant features based on sensitivity analysis enhances model performance



P. Shrestha and I. Savidis, "EDA-ML: Graph representation learning framework for digital IC design automation," *Proceedings of the International Symposium on Quality Electronic Design (ISQED)*, pp. 241–246, April 2024

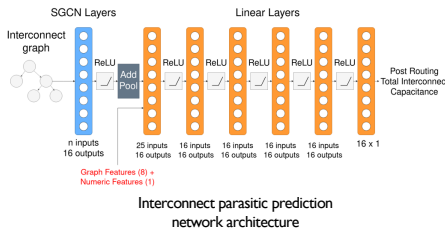
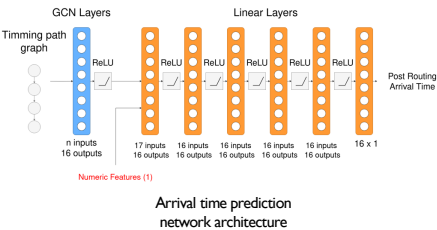
173

Applications

174

- Two distinct downstream metric prediction problems are used as case studies

Problem Formulation	Metric to predict (M)	Arrival Time Prediction	Interconnect Parasitic Prediction
	Initial phase (X)	Gate arrival time	Total interconnect capacitance
	Final phase (Y)	Post logical synthesis	Post placement
	Graph representation (G)	Timing path graph (TPG)	Interconnect graph (IG)
	Prediction level	Node level	Graph level
Network Architecture	Baseline (M_b)	Post logical synthesis arrival time	Post placement total interconnect capacitance
	Graph embedding layer	GCN	SGCN
	Use of graph pooling	No	Yes
Model Training and Evaluation Details	Loss function (L)	Mean Squared Error Loss (MSE Loss)	
	Optimizer (O)	Stochastic Gradient Descent (SGD) [13]	
	Learning Rate (LR)	1%	
	Batch Processing (B)	Yes (1024 batches per iteration)	
	Evaluation metrics	MAE, MAPE, worse 1% MAE, worse 1% MAPE	



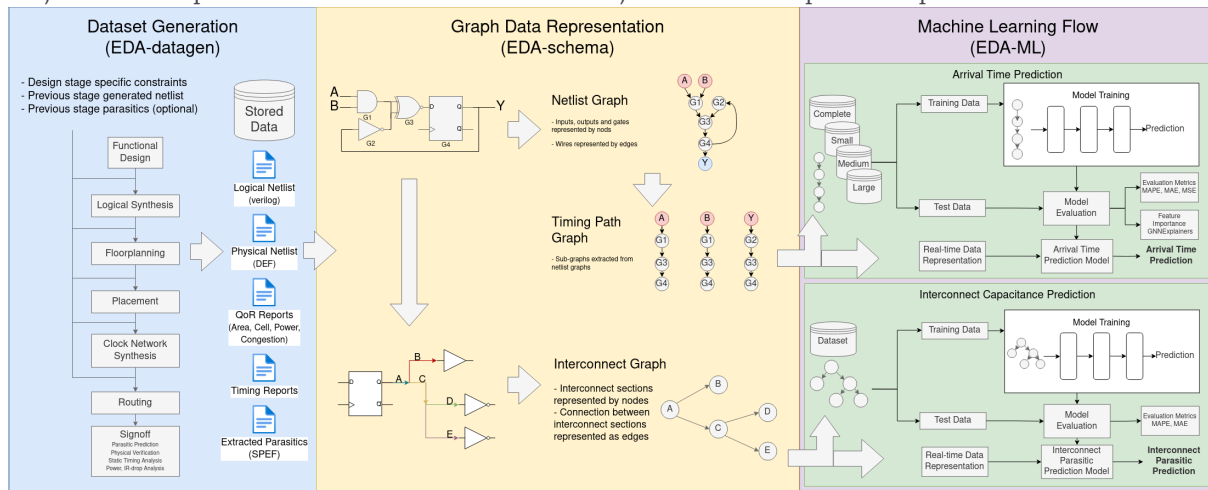
P. Shrestha and I. Savidis, "EDA-ML: Graph representation learning framework for digital IC design automation," *Proceedings of the International Symposium on Quality Electronic Design (ISQED)*, pp. 241–246, April 2024

174

- Current research based on this flow are

a) arrival time prediction

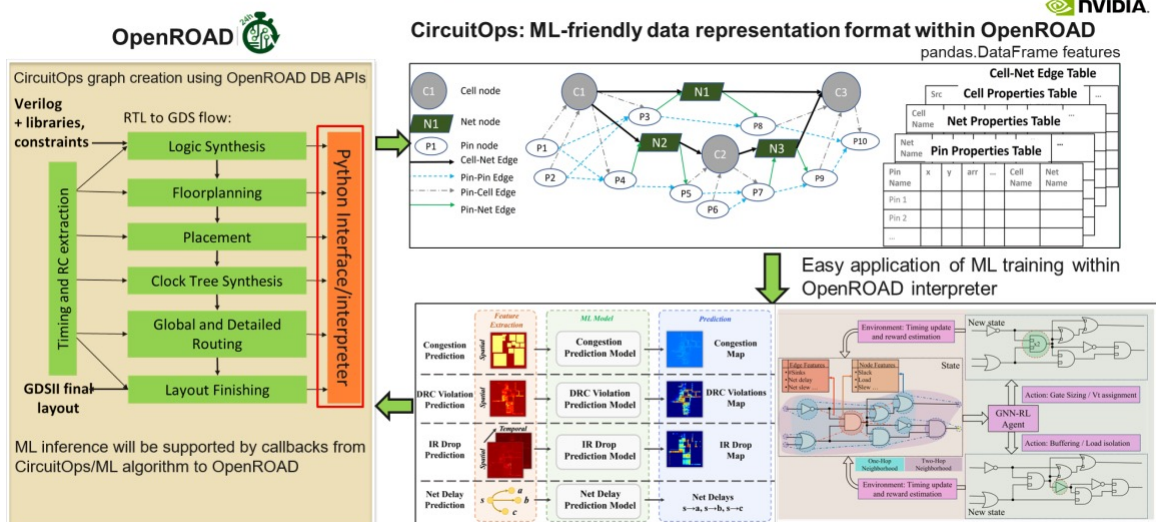
b) interconnect capacitance prediction



P. Shrestha and I. Savidis, "EDA-ML: Graph representation learning framework for digital IC design automation," *Proceedings of the International Symposium on Quality Electronic Design (ISQED)*, pp. 241–246, April 2024

175

OpenROAD as an ML for Chip Design Playground



Source: <https://vlsicad.ucsd.edu/NEWS24/IITG-Kahng-v3.pptx>

176

Outline of Presentation

177

- Introduction to Electronic Design Automation
- Machine learning techniques
- Case studies
- Standardizing ML for digital EDA
- **Conclusions**

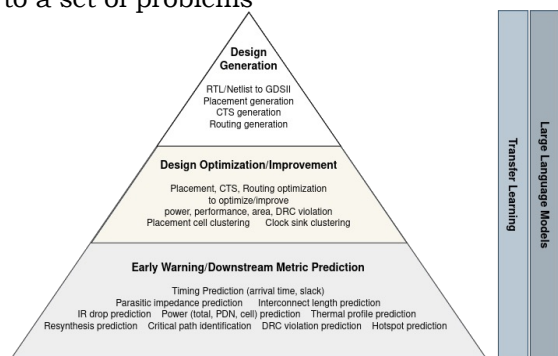


177

Summary of AI-driven Design Automation

178

- ML applied to the design space can be generalized to a set of problems
 - Downstream Metric Prediction
 - Circuit component optimization
 - Circuit component generation
- Key circuit representations
 - Image based
 - Graph based
 - Tabular based
- Benefits brought by ML for EDA
 - Reduced simulations required, reduced turnaround time
 - Design space exploration
 - Prediction of parasitic impedances, reliability and variability
 - Guide optimization or direct generation of schematic and layout design
 - Migration and reuse of pre-trained models



178

Future Directions

179

- Improve reliability, robustness, and interpretability of ML models for EDA
- Standardized Machine Learning
 - Standardized benchmarks and applications
 - Standardized ML flow
 - Common evaluation protocols
- Meta-learning
 - Learning what to learn
 - Learn parameter values for base (pre-trained) models for circuit tasks
 - Learning which model to learn
 - Auto select the ML and optimization algorithms best suited for a given circuit task
 - Learning how to learn
 - Auto hyperparameter tuning of ML models and generation of pipeline for EDA



179

Future Directions

180

- Open Datasets and Leaderboards

The screenshot shows the Hugging Face website interface. On the left, there's a sidebar with 'Tasks' and 'Datasets' sections. The 'Tasks' section lists various tasks like Depth Estimation, Image Classification, Object Detection, Image Segmentation, Text-to-Image, Image-to-Text, Image-to-Image, Image-to-Video, Unconditional Image Generation, Video Classification, Text-to-Video, Zero-Shot Image Classification, Mask Generation, and Zero-Shot Object Detection. The 'Datasets' section shows a list of datasets including H-D-T/Buzz, m-a-p/Matrix, HuggingFaceFW/fineweb, allenai/WildChat-1M, and TIGER-Lab/MMLU-Pro. On the right, there's a section titled 'Open LLM Leaderboard' which includes an 'LLM Benchmark' table. The table has columns for 'T', 'Model', 'Average', and 'AR'. The data rows are as follows:

T	Model	Average	AR
◆	davidkim295/Rhea-72b-v0.5	81.22	79
#	Mistral/Mistral-70B	81	78
#	Mistral/Mistral-70B	80.98	78
◆	abacusai/Sage-72b-v0.1	80.48	76
◆	ibivibiv/alpaca-dragon-72b-v1	79.3	73
#	mistralai/Mistral-8x22B-Instruct-v0.1	79.15	72
#	MaziyarPanahi/Llama-3-70B-Instruct-DPO-v0.2	78.96	72
#	MaziyarPanahi/Llama-3-70B-Instruct-DPO-v0.4	78.89	72

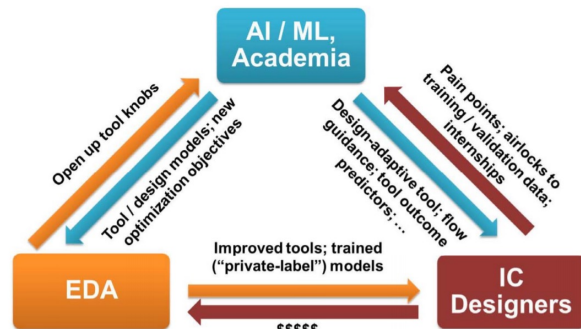


180

Conclusions

181

- EDA tools are like autonomous vehicles
 - Currently, driver control/attention is still required
- EDA tools are like autonomous vehicles to be driven in more challenging road conditions
- Level of automation will keep rising
- More collaborations needed between circuit design, academia and industry



A. B. Kahng, "Machine Learning Applications in Physical Design: Recent Results and Directions", *Proceedings of the International Symposium on Physical Design (ISPD)*, pp. 68-73, March 2018

181



<http://ice.ece.drexel.edu>



183

References

184

- A. Sangiovanni-Vincentelli, "The Tides of EDA", *IEEE Journals & Magazine*, Vol. 20, No. 6, pp. 59-75, Dec. 2003
- O. Aydin, and O. Uçar, "Design for Smaller, Lighter and Faster ICT Products: Technical Expertise, Infrastructures and Processes." *Advances in Science, Technology and Engineering Systems Journal*, Vol. 2, No. 3, pp. 1114-1128, July 2017.
- A. Moubayed, M. Injadat, A. B. Nassif, H. Lutfiyya and A. Shami, "E-Learning: Challenges and Research Opportunities Using Machine Learning & Data Analytics," *IEEE Access*, Vol. 6, No.1, pp. 39117-39138, July 2018
- K. Li, J. Malik, "Learning to optimize," *Proceedings of the International Conference on Learning Representations (ICLR)*, pp. 1-13, April 2017
- Y.-C. Lu, J. Lee, A. Agnesina, K. Samadi, and S. K. Lim. "GAN-CTS: A generative adversarial framework for clock tree prediction and optimization," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1-8, Nov. 2019
- Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip. "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 32, No. 1, pp. 4-24, Jan. 2020.
- J. K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How Powerful are Graph Neural Networks?," *Proceedings of the International Conference on Learning Representations (ICLR)*, pp. 1-17, Dec 2018
- T. Danel, P. Spurek, J. Tabor, M. Smieja, L. Struski, A. Słowik, and L. Maziarka, "Spatial graph convolutional networks," *Proceedings of the International Conference on Neural Information Processing (NIPS)*, pp. 668-675, Nov. 2020
- I. Goodfellow, et al., "Generative Adversarial Nets", *Proceedings of the International Conference on Neural Information Processing System (NIPS)*, Vol. 2, No.1, pp.2672-2680, Dec. 2014
- D. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *Proceedings of the International Conference on Learning Representations (ICLR)*, pp.1-14, Nov 2013
- D. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *Proceedings of the International Conference on Learning Representations (ICLR)*, pp.1-14, Nov 2013
- V. A. Chhabria, W. Jiang, A. B. Kahng and S. S. Sapatnekar, "From Global Route to Detailed Route: ML for Fast and Accurate Wire Parasitics and Timing Prediction," *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, pp. 7-14, Sept. 2022
- P. Shrestha, S. Phatharodom, and I. Savidis, "Graph representation learning for gate arrival time prediction," *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, pp. 127-133, Sept. 2022



184

References

185

- G. Pradipta, V. A. Chhabria, and S. S. Sapatnekar, "A Machine Learning Based Parasitic Extraction Tool," *Workshop on Open-Source EDA Technology (WOSET)*, pp. 1-3, Nov. 2019
- H. Ren, G. F. Kokai, W. J. Turner and T. Ku. "ParaGraph: Layout Parasitics And Device Parameter Prediction Using Graph Neural Networks," *Proceedings of the ACM/IEEE Design Automation Conference (DAC)*, pp. 1-6, June 2020
- P. Shrestha and I. Savidis, "Graph Representation Learning for Parasitic Impedance Prediction of the Interconnect" *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1-5, May 2023
- Y.-C. Lu, W.-T. Chan, V. Khandelwal, and S. K. Lim, "Driving Early Physical Synthesis Exploration through End-of-Flow Total Power Prediction," *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, pp. 97-102, Sept. 2022
- Y.-C. Lu, T. Yang, S. K. Lim, and H. Ren, "Placement optimization via PPA-directed graph clustering", *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, pp. 1-6, Sept. 2022
- A. Mirhoseini, et al., "A Graph Placement Methodology for Fast Chip Design", *Nature*, No. 594, pp. 207-212, June 2021
- H. Liao, W. Zhang, X. Dong, B. Poczoz, K. Shimada, & L. B. Kara, "A deep reinforcement learning approach for global routing." *Journal of Mechanical Design*, Vol. 142, No. 6, pp 061701, Nov. 2019
- K. Zhu. et al., "GeniusRoute: A New Analog Routing Paradigm Using Generative Neural Network Guidance," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp.1-8, Nov 2019
- S. S. Han, A. B. Kahng, S. Nath, & A. S. Vydyanathan, "A deep learning methodology to proliferate golden signoff timing", *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1-6, March 2014
- P. Shrestha and I. Savidis, "Transfer Learning of Arrival Time Prediction Models from a 65 nm to a 28 nm Process Node", *Proceedings of the IEEE Dallas Circuits and Systems Conference (DCAS)*, pp. 1-6, April 2024
- Z. Wu and I. Savidis, "Transfer Learning for Reuse of Analog Circuit Sizing Models Across Technology Nodes," *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1-5, May 2022
- L. Mingjie, N. Pinckney, B. Khailany, and H. Ren, "VerilogEval: Evaluating large language models for verilog code generation," *Proceedings of the IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 1-8, Oct 2023
- L. Mingjie, T.-D. Ene, R. Kirby, C. Cheng, N. Pinckney, R. Liang, J. Alben et al., "ChipNeMo: Domain-Adapted LLMs for Chip Design", preprint arXiv:2311.00176 , pp. 1-23, Apr 2024



185

References

186

- N. Pinckney, C. Deng, C.-T. Ho, Y.-D. Tsai, M. Liu, W. Zhou, B. Khailany, and H. Ren, "Comprehensive Verilog Design Problems: A Next-Generation Benchmark Dataset for Evaluating Large Language Models and Agents on RTL Design and Verification.", arXiv preprint arXiv:2506.14074, pp. 1–16, Jun 2025
- U. Sharma, B.-Y. Wu, S. R. D. Kankipati, V. A. Chhabria, and A. Rovinski, "Openroad-assistant: An open-source large language model for physical design tasks.", In Proceedings of the ACM/IEEE International Symposium on Machine Learning for CAD (MLCAD), pp. 1–7, Sep 2024
- M. Shalan and T. Edwards, "Building OpenLANE: A 130nm OpenROAD-based Tapeout-Proven Flow", *Proceedings of the IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 1–8, Nov 2020
- P. Shrestha, A. Aversa, S. Phatharodom, and I. Savidis, "EDA-schema: A graph datamodel schema and open dataset for digital design automation", *Proceedings of the ACM Great Lakes Symposium on VLSI (GLSVLSI)*, pp. 1–8, June 2024
- A. Agnesina, K. Chang, and S. K. Lim, "VLSI placement parameter optimization using deep reinforcement learning," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–9, Nov. 2020
- Z. Xie, R. Liang, X. Xu, J. Hu, Y. Duan, and Y. Chen, "Net2: A graph attention network method customized for pre-placement net length estimation," *Proceedings of the Asia and South Pacific Design Automation Conference (ASPDAC)*, pp. 671–677, Jan. 2021
- E. C. Barboza, N. Shukla, Y. Chen, and J. Hu, "Machine learning-based pre-routing timing prediction with reduced pessimism," *Proceedings of the Design Automation Conference (DAC)*, pp. 1–6, Jun. 2019
- P. Shrestha and I. Savidis, "EDA-ML: Graph representation learning framework for digital IC design automation", *Proceedings of the International Symposium on Quality Electronic Design (ISQED)*, pp. 241–246, April 2024
- A. B. Kahng, "Machine Learning Applications in Physical Design: Recent Results and Directions", *Proceedings of the International Symposium on Physical Design (ISPD)*, pp. 68–73, March 2018
- S. Mohamadi, G. Mujtaba, N. Le, G. Doretto, D. A. Adjeroh, "ChatGPT in the Age of Generative AI and Large Language Models: A Concise Survey", arXiv preprint arXiv:2307.04251, pp.1–60, July 2023
- N. Pinckney, C. Deng, C.-T. Ho, Y.-D. Tsai, M. Liu, W. Zhou, B. Khailany, and H. Ren, "Comprehensive Verilog Design Problems: A Next-Generation Benchmark Dataset for Evaluating Large Language Models and Agents on RTL Design and Verification.", arXiv preprint arXiv:2506.14074, pp. 1–16, Jun 2025

