UNIVERSITY of GUELPH

GUELPH ENGINEERING

A Deep Learning Framework to Predict Routability for FPGA Circuit Placement

A. Al-Hyari, A. Shamli, Z. Abuowaimer, G. Grewal, and S. Areibi

Guelph FPGA CAD Group

https://fpga.socs.uoguelph.ca

UNIVERSITY of GUELPH School of Computer Science

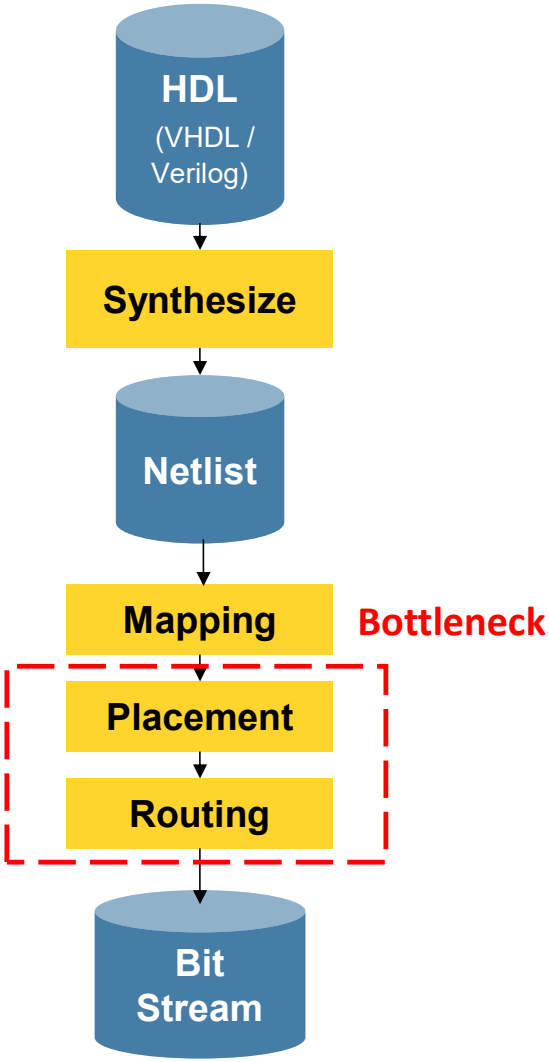
Motivation

Two major problems:

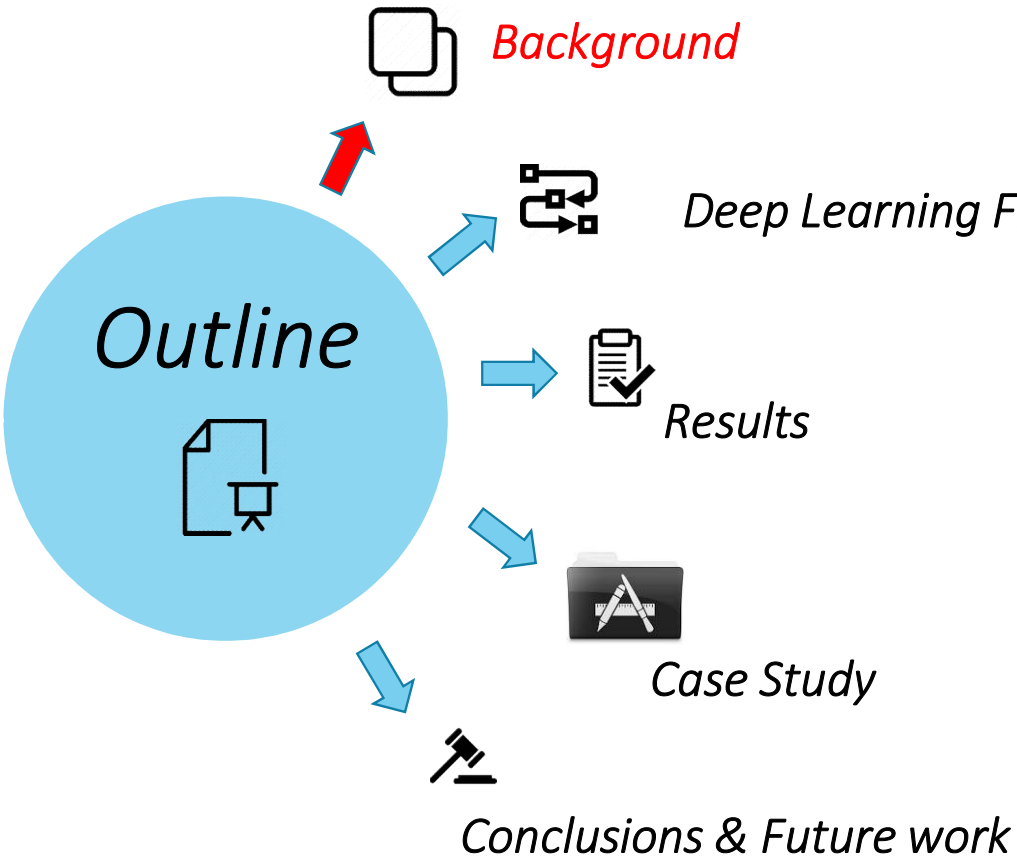
- Place-and-Route are the two most time-consuming steps in the FPGA CAD flow
- Placement solutions produced by a placement tool may be unroutable.

Fast and Accurate Routability Prediction:

- Ability for placer to respond early and often to improve P&R runtimes and increase likelihood of producing a routable placement

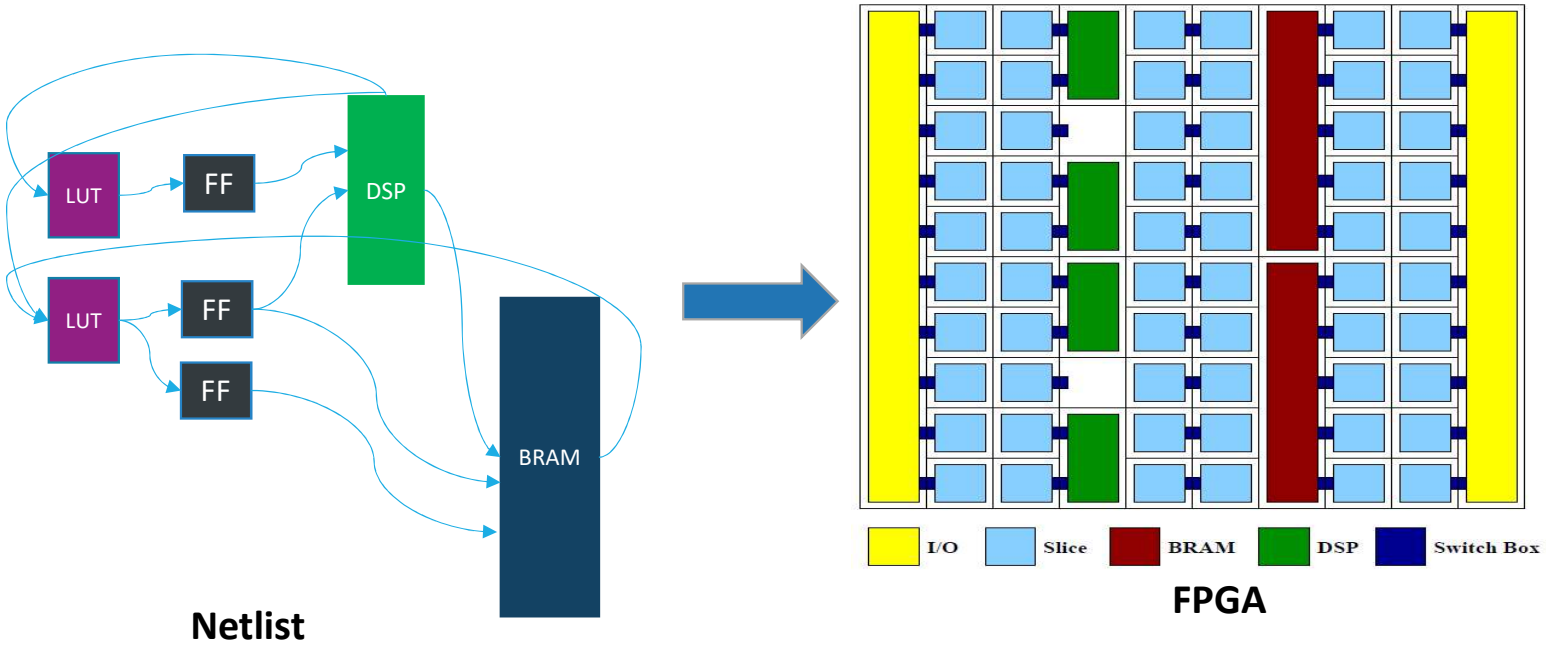


Outline


Outline
Background
Deep Learning Framework
Results
Case Study
Conclusions & Future work

Placement Problem

- Given a circuit in the form of a netlist, map the components in the netlist onto locations (resources) on the FPGA such that:
  - Minimize objectives ➔ wirelength, delay, congestion, etc
  - Subject to several constraints: based on architecture of FPGA
- Target: Xilinx Ultrascale

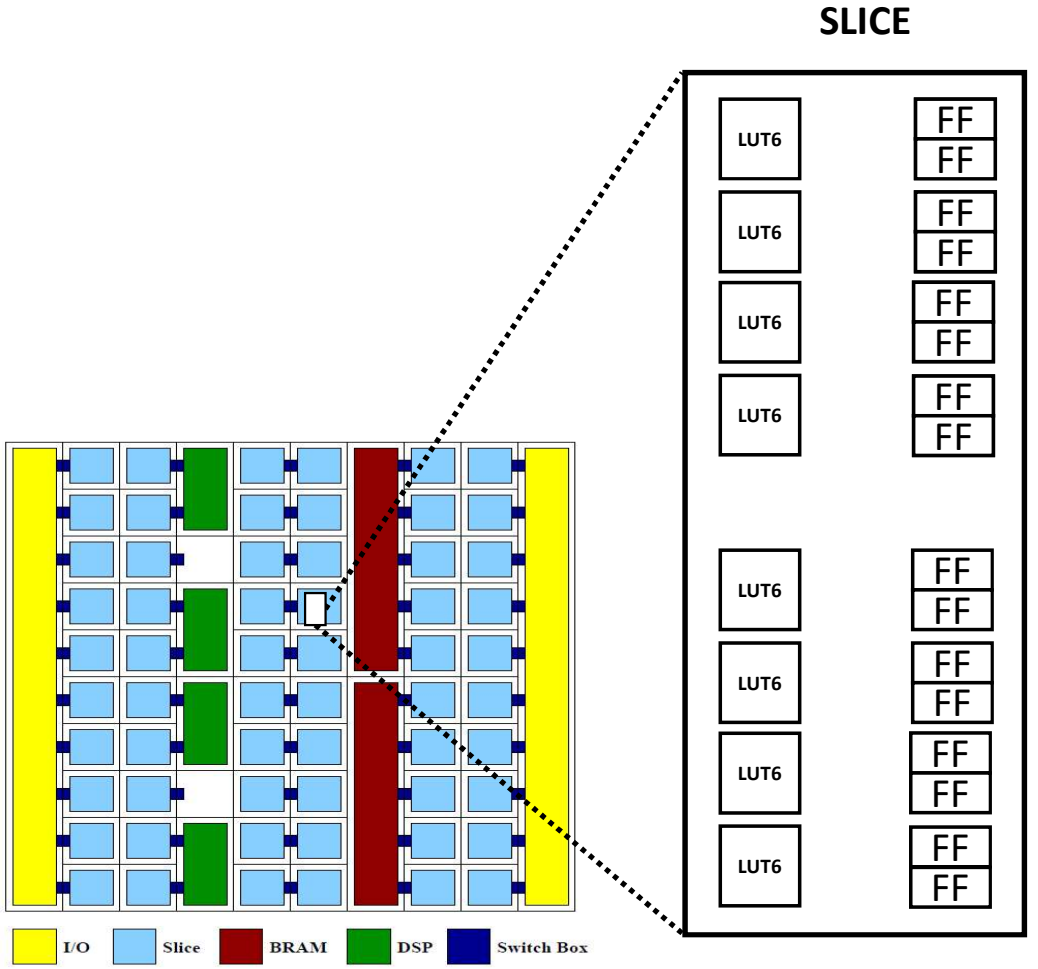


Netlist

FPGA

Modern FPGA Architecture

- Architecture of modern FPGA devices imposes additional constraints on placement problem
- Modern FPGAs are heterogeneous
- Slice architecture imposes constraints on packing and placement



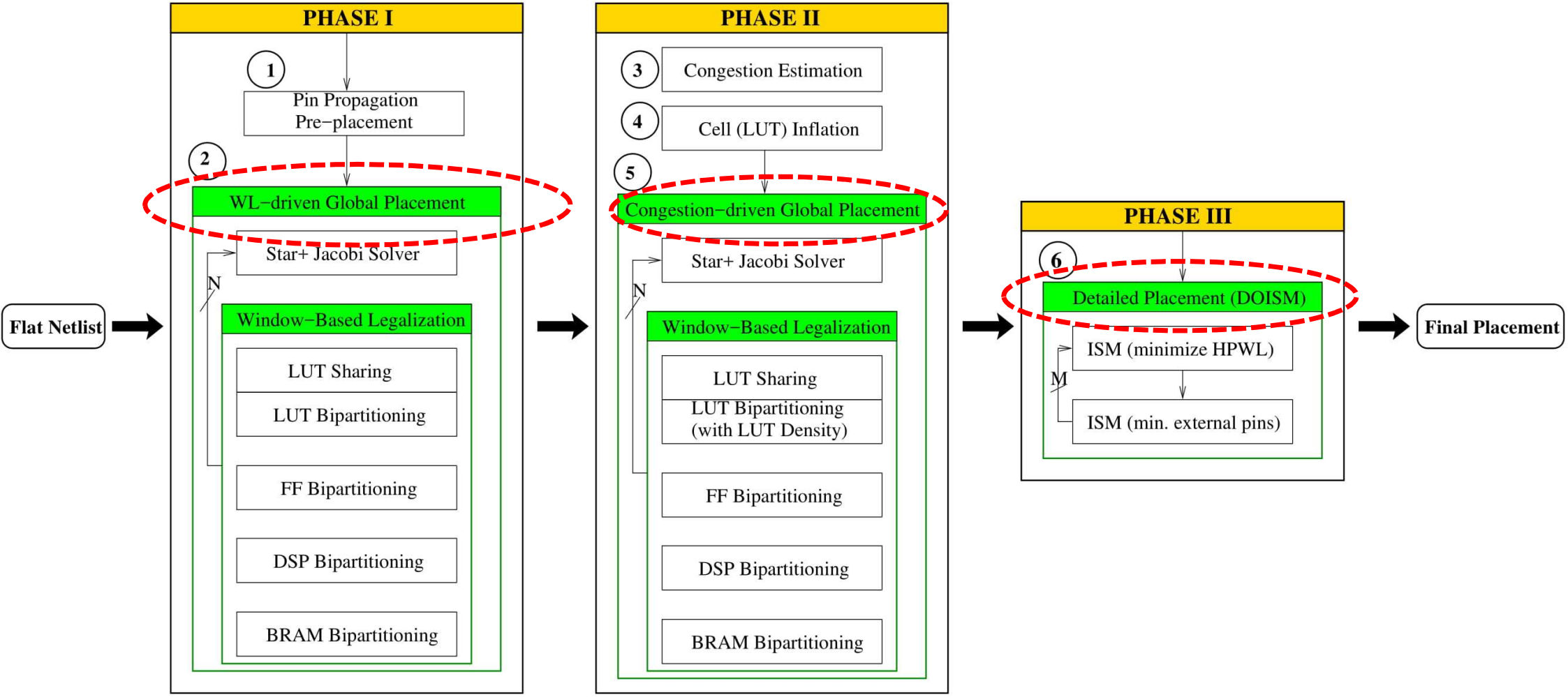
Benchmarks

Design	#LUTs (util)	#Flops (util)	#BRAMs	#DSPs	#control sets	Rent Exponent
FPGA-1	50K (9%)	55K (5%)	0 (0%)	0 (0%)	12	0.4
FPGA-2	100K (19%)	66K (6%)	100 (6%)	100 (13%)	121	0.4
FPGA-3	250K (47%)	170K (16%)	600 (35%)	500 (65%)	1281	0.6
FPGA-4	250K (47%)	172K (16%)	600 (35%)	500 (65%)	1281	0.7
FPGA-5	250K (47%)	174K (16%)	600 (35%)	500 (65%)	1281	0.8
FPGA-6	350K (65%)	352K (33%)	1000 (58%)	600 (78%)	2541	0.6
FPGA-7	350K (65%)	355K (33%)	1000 (58%)	600 (78%)	2541	0.7
FPGA-8	500K (93%)	216K (20%)	600 (35%)	500 (65%)	1281	0.7
FPGA-9	500K (93%)	366K (34%)	1000 (58%)	600 (78%)	2541	0.7
FPGA-10	350K (65%)	600K (56%)	1000 (58%)	600 (78%)	2541	0.6
FPGA-11	480K (89%)	363K (34%)	1000 (58%)	400 (52%)	2091	0.7
FPGA-12	500K (93%)	602K (56%)	600 (35%)	500 (65%)	1281	0.6

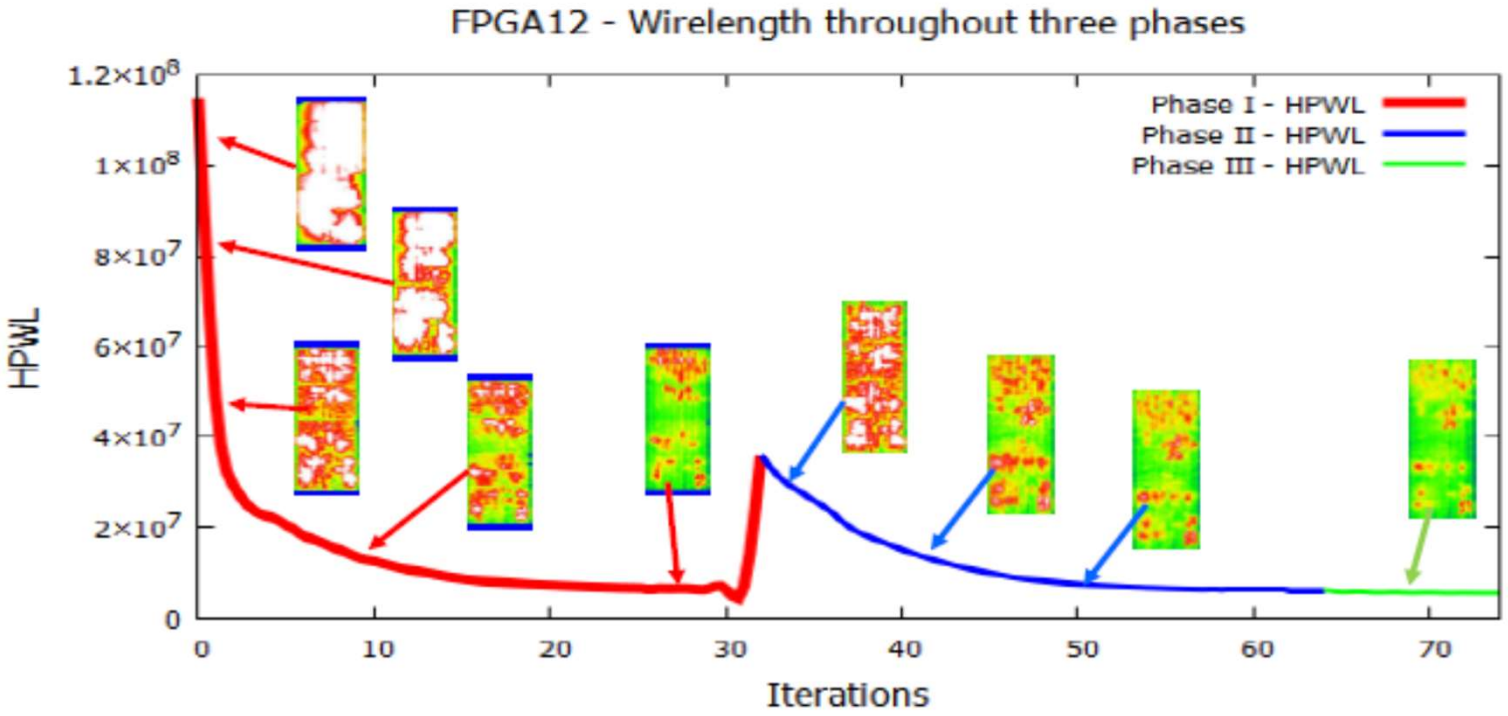
- The 12 ISPD 2016 routing-aware placement contest circuits
- Our industrial partner, Xilinx Inc., synthesized an extra 360 benchmarks using an internal netlist generation tool

#LUTs	#FFs	#BRAMs	#DSPs	#CSETs	#IOs	Rent Exp
44K – 518K	52K – 630K	0 - 1035	0 - 620	11 - 2684	150 - 600	0.4 – 0.8

GPlace3.0 Flow

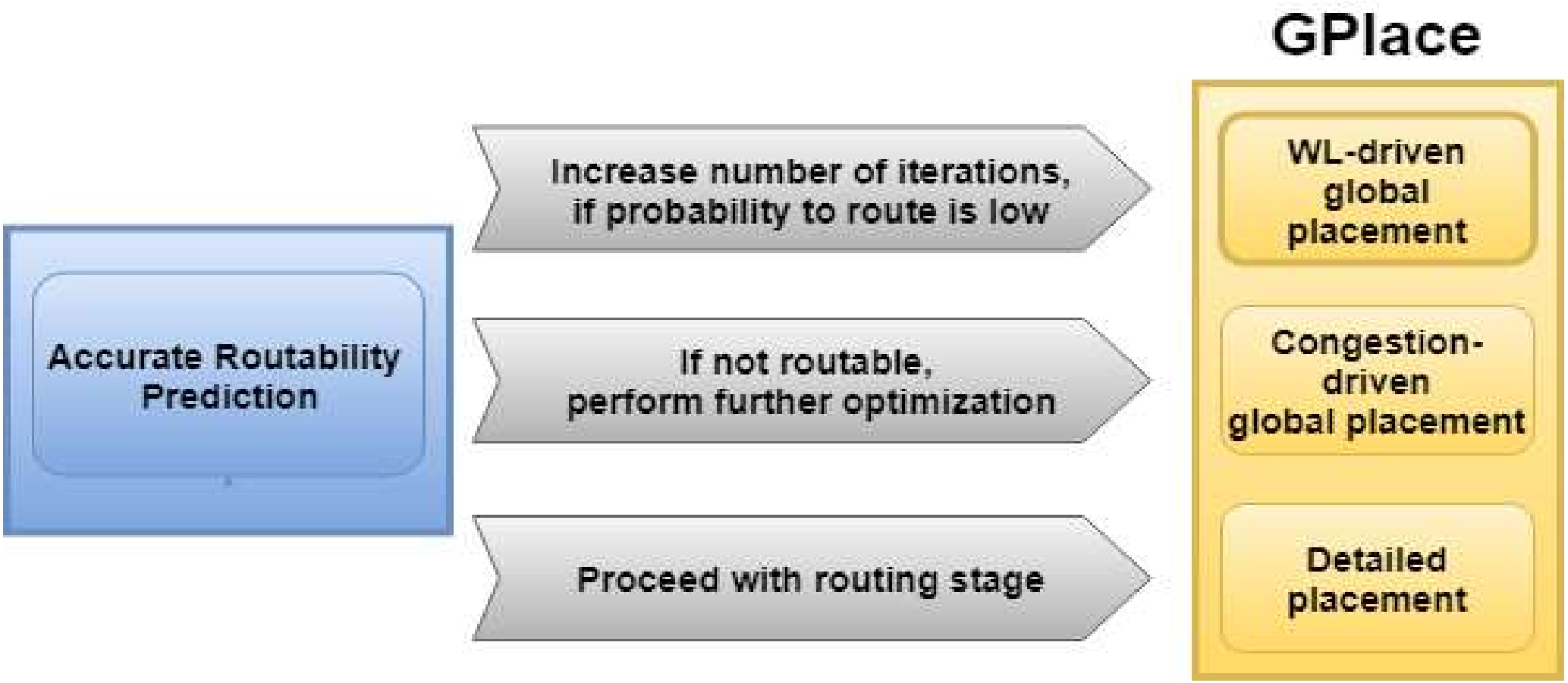


Congestion & WL across 3-phases of Placement



GPlace3.0 minimizes both wirelength and congestion over time, but without any knowledge of whether or not the solution is routable or not

Benefits of Predicting Routability

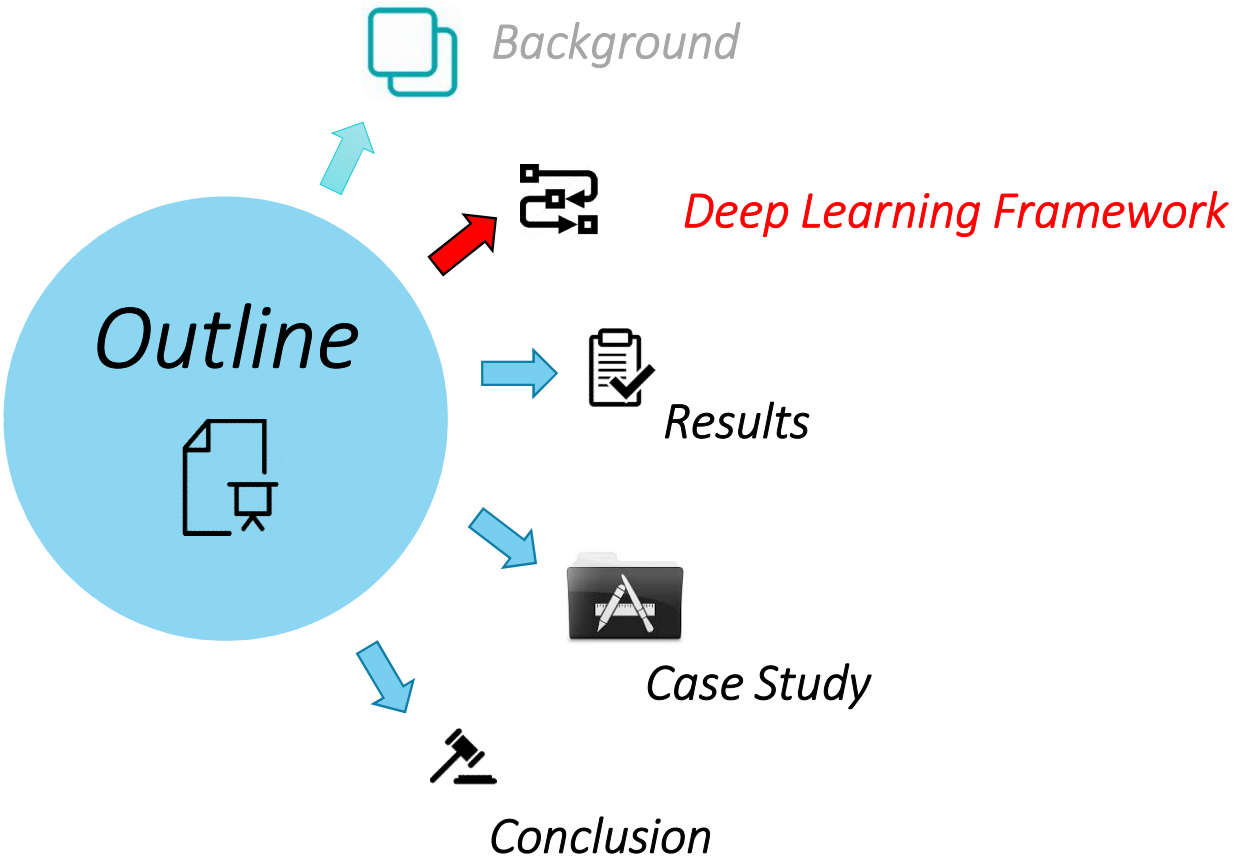


Deep Learning & EDA

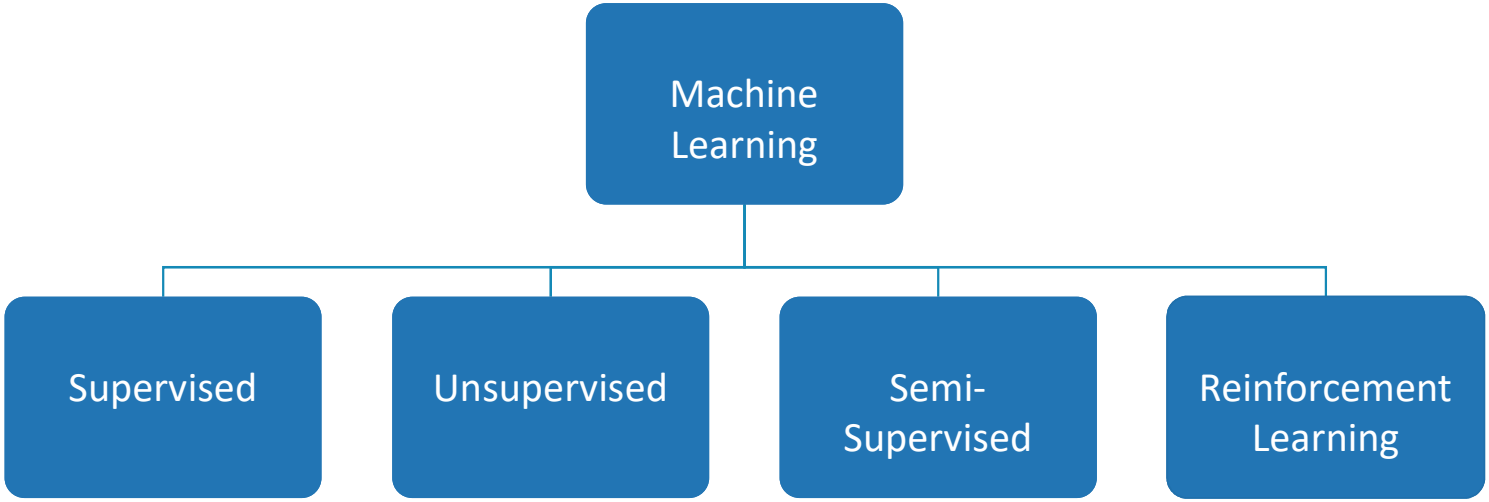
- The ability to accurately and efficiently estimate the routability of a circuit based on its placement is one of the most challenging tasks in the FPGA flow.
- Providing an informative feedback about the routability can help the placement tool to further enhance its optimization strategy.

- **Questions:**
  - **Is it possible to develop a framework that is placer independent and architecture independent?**
  - **Can Deep Learning be used to predict the routability of a placement?**

Outline for Reminder of Talk

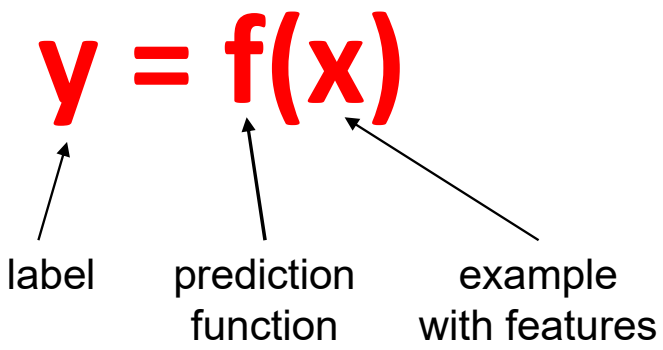


Machine Learning



Machine learning is a type of Artificial Intelligence (AI) that can provide systems with the ability to learn without being explicitly programmed.

Supervised Machine Learning

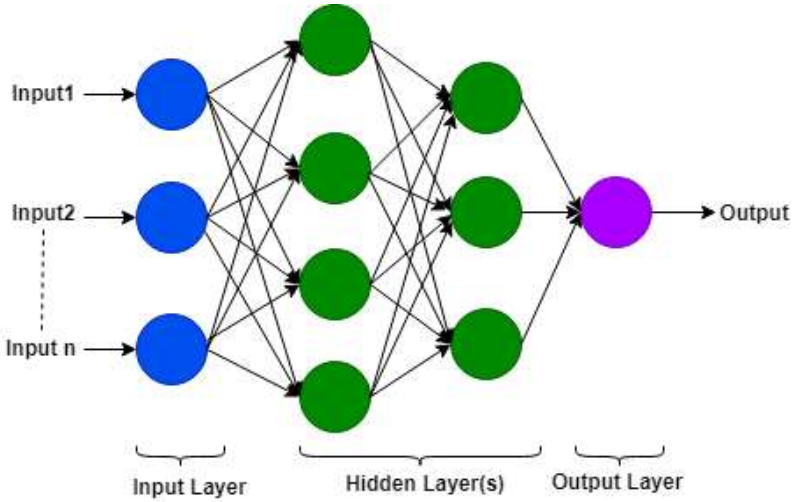


- **Training:** given a *training set* of labeled examples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, estimate the prediction function f() by minimizing the prediction error on the training set

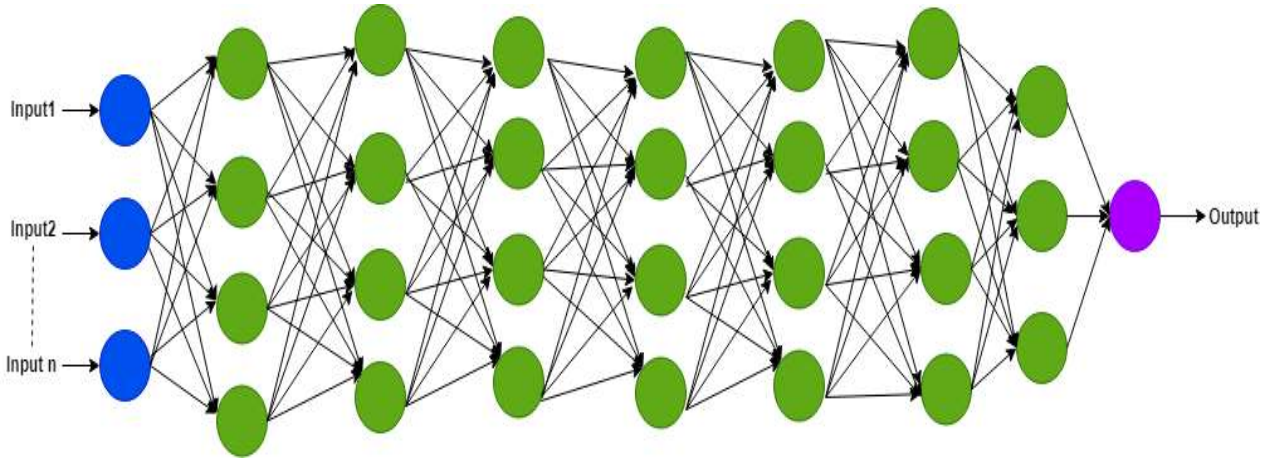
- **Testing:** apply f() to a never before seen *test example* **x** and output the predicted value $y = f(\mathbf{x})$

Deep Learning

- Deep learning (DL) is a subset of machine learning, **it refers to having deeper hierarchy in a neural network**.
- **DL is capable of processing high dimensional data**.



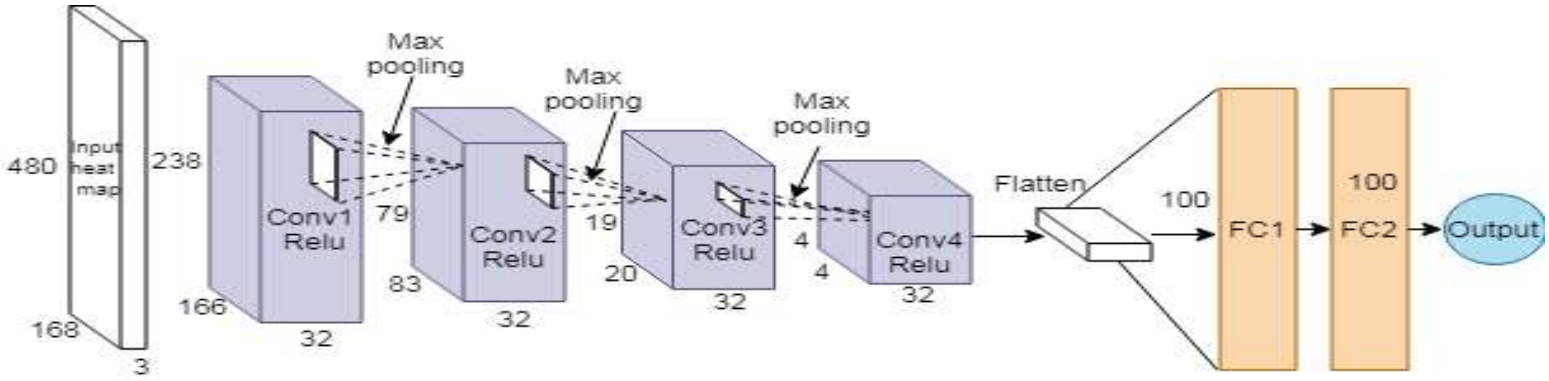
Shallow network



Deep network

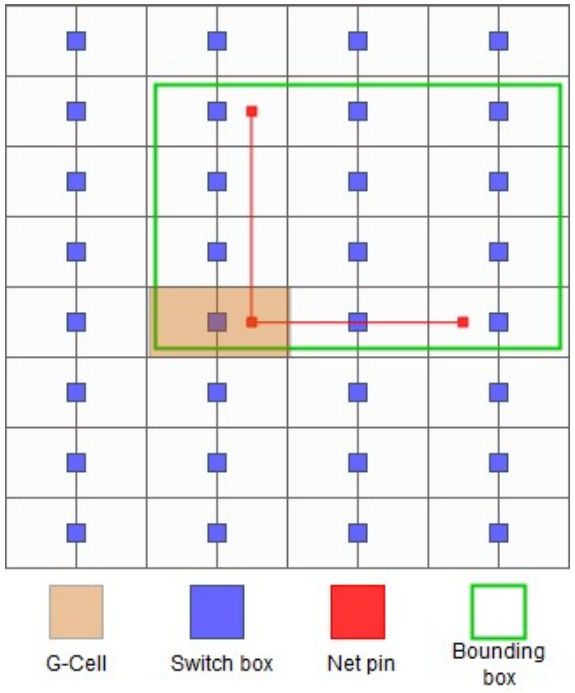
Convolutional Neural Network (CNN)

- CNN is the most commonly used form of DL because it can process an image and generate a meaningful response depending on the application.
- Convolutional filters are capable of capturing the spatial relationship between surrounding elements implicitly.



Features

- Four features are calculated for each G-Cell (corresponds to a switch box) of the FPGA
- Each feature is designed to characterize the routing resource utilization of the switch



f_1 = WLPA

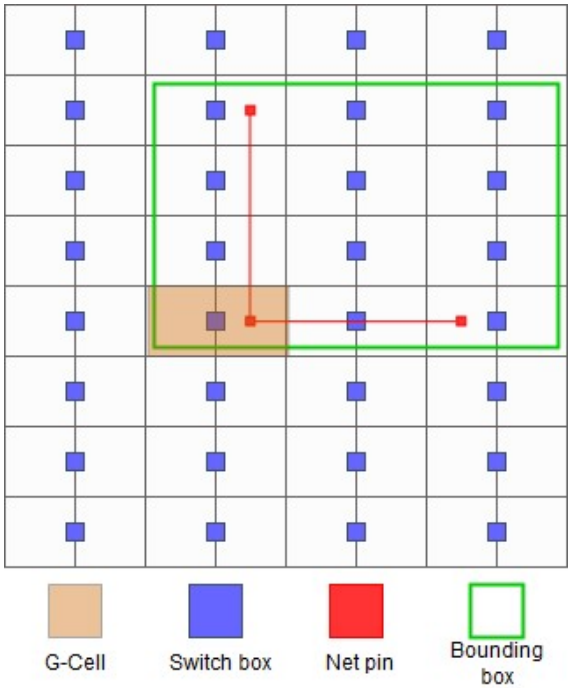
f_2 = Pin density

f_3 = $NCPR_{5x5}$

f_4 = $NCPR_{9x9}$

Features

- Four features are calculated for each G-Cell (corresponds to a switch box) of the FPGA
- Each feature is designed to characterize the routing resource utilization of the switch



$$f_1 = \sum_{n \in N_i} \frac{w_n . HPWL_n}{\#gcell_n}$$

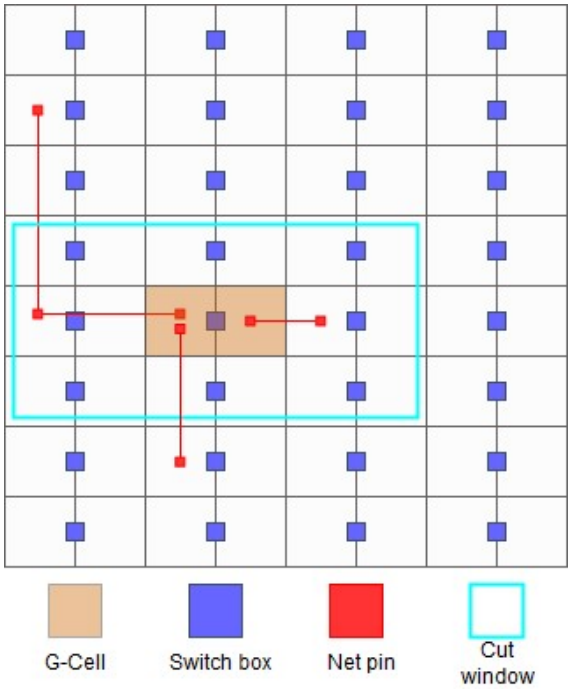
$w_n = 1$

$HPWL_n = 5$

$\#gcell_n = 12$

Features

- Four features are calculated for each G-Cell (corresponds to a switch box) of the FPGA
- Each feature is designed to characterize the routing resource utilization of the switch



$$f_1 = \sum_{n \in N_i} \frac{w_n . HPWL_n}{\#gcell_n}$$

$$f_2 = \sum_{n \in N_i} \#pins_{n,gcell_i}$$

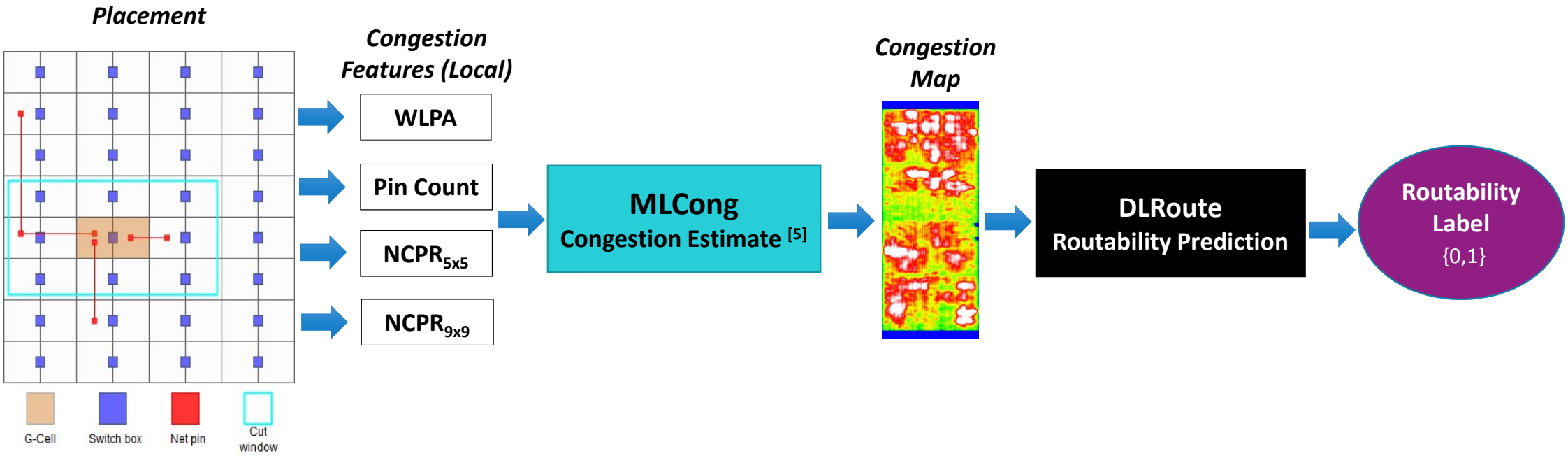
$$f_3 = | W_{5x5} |$$

$$f_4 = | W_{9x9} |$$

$\#pins_{n,gcel\ i} = 3$

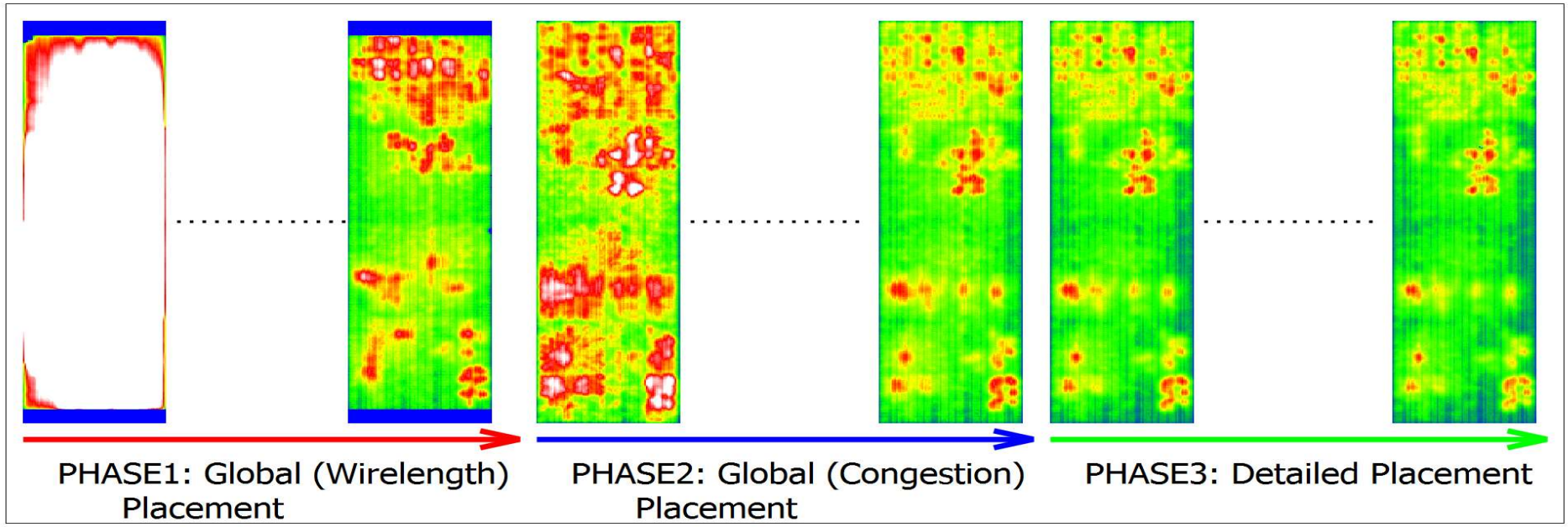
$|W_{3x3}| = 2$

DLRoute: Overall Methodology

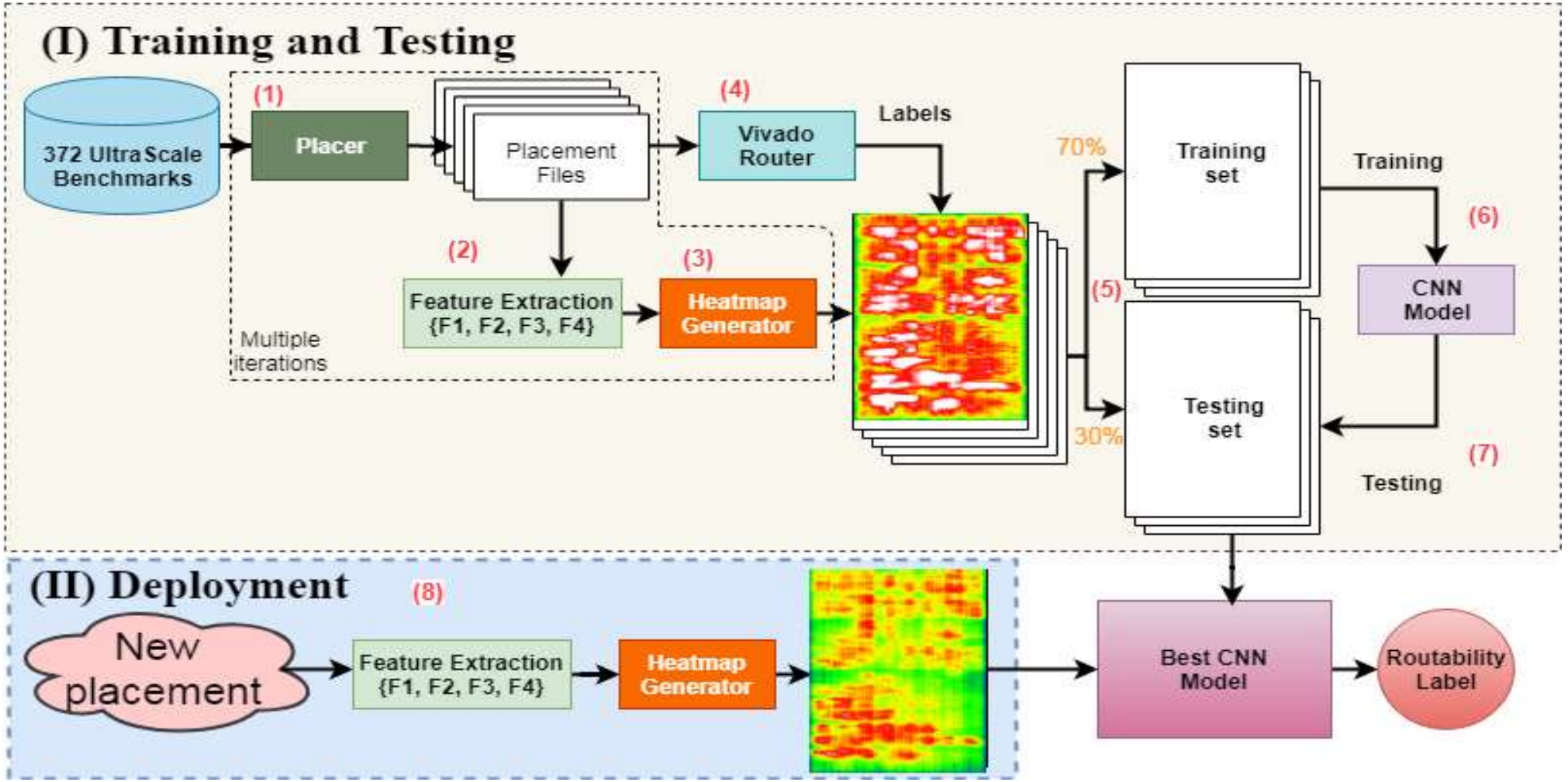


DLRoute: Data Collection

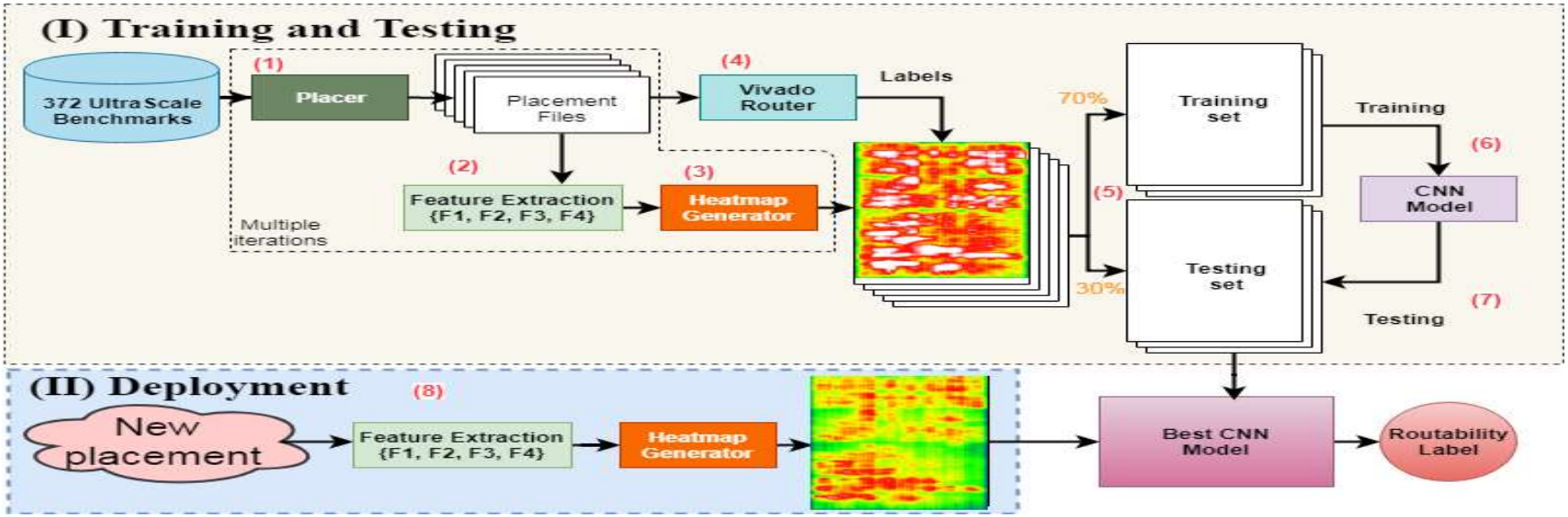
Heat-maps change across the three phases of placement flow



DLRoute Framework

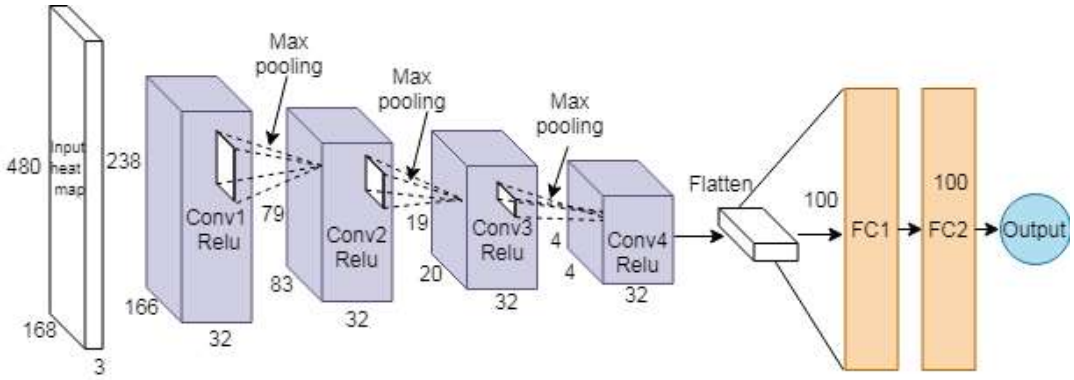


DLRoute Framework for Routability Prediction



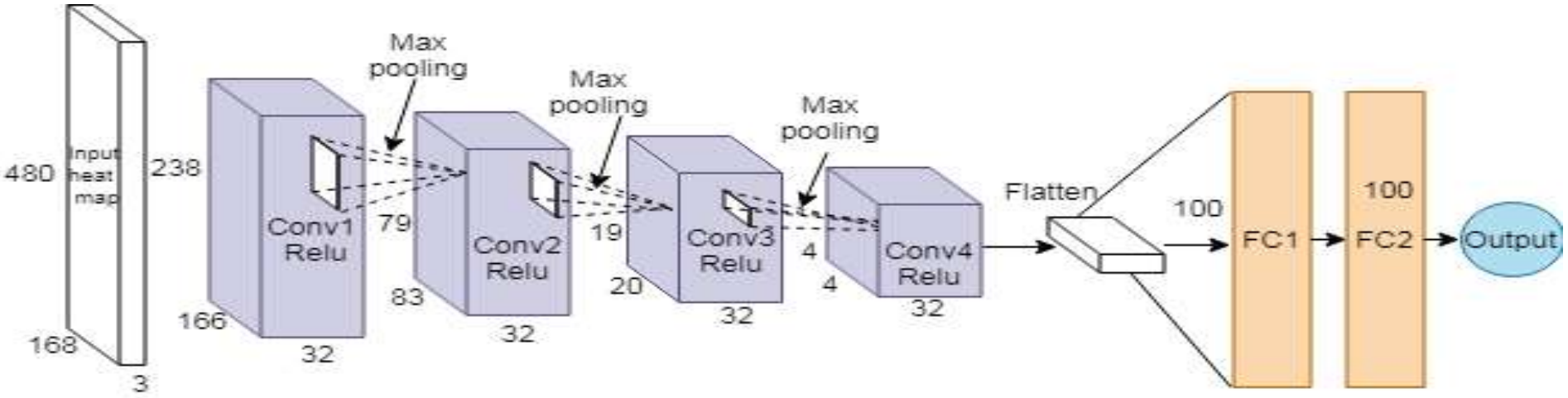
- [illegible]
- [illegible]
- [illegible]
- [illegible]

CNN Architecture



- [illegible]
- [illegible] is applied to the output of CONV2 layer
- The activation function is ReLU
- The generated feature maps are 4x4 with 32 dimension

DLRoute: CNN Architecture



- The network takes a congestion heat map of size 480x168
- Four convolutional layers with a depth of 32 filters are used to extract features
- Two fully connected layers are used to classify the flattened vector of features
- A sigmoid output neuron generates a binary label of {0, 1} as routability label

DLRoute: Architecture Details

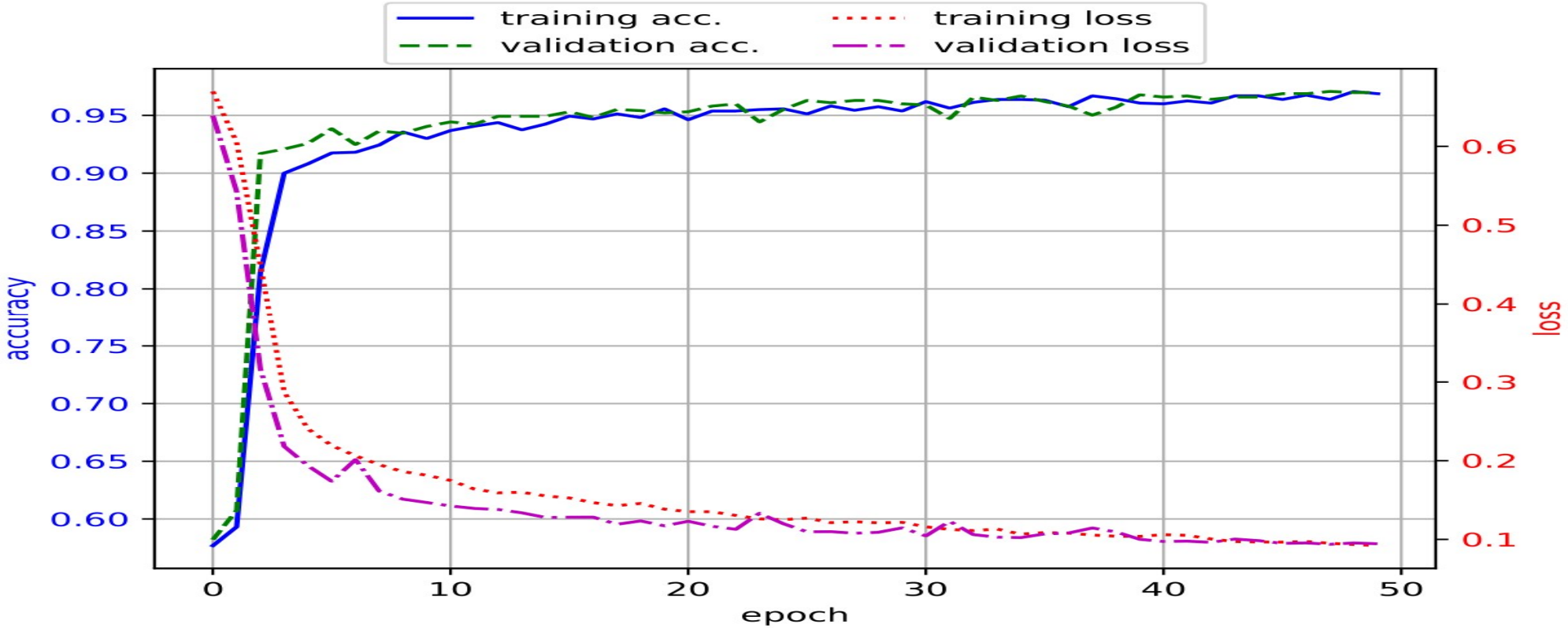
Optimized CNN Architecture

#	Architecture Parameters	Output Volume	# of Parameters
1	CONV1: 32x(6,3), stride: (2,1)	238x166x32	1760
2	Maxpool: (3,2), stride: (3,2)	79x83x32	0
3	CONV2: 32x(3,3), stride: (2,2)	39x41x32	9248
4	Maxpool: (2,2), stride: (2,2)	19x20x32	0
5	CONV3: 32x(3,3), stride: (2,2)	9x9x32	9248
6	Maxpool: (2,2), stride: (2,2)	4x4x32	0
7	CONV4: 32x(3,3), stride: (1,1)	2x2x32	9248
8	Flatten	128	0
9	FC1: 100 (ReLU)	100	12900
10	FC2: 100 (ReLU)	100	10100
11	OUT: 1 (Sigmoid)	1	101
Total number of parameters			52605

Best Hyper-Parameters

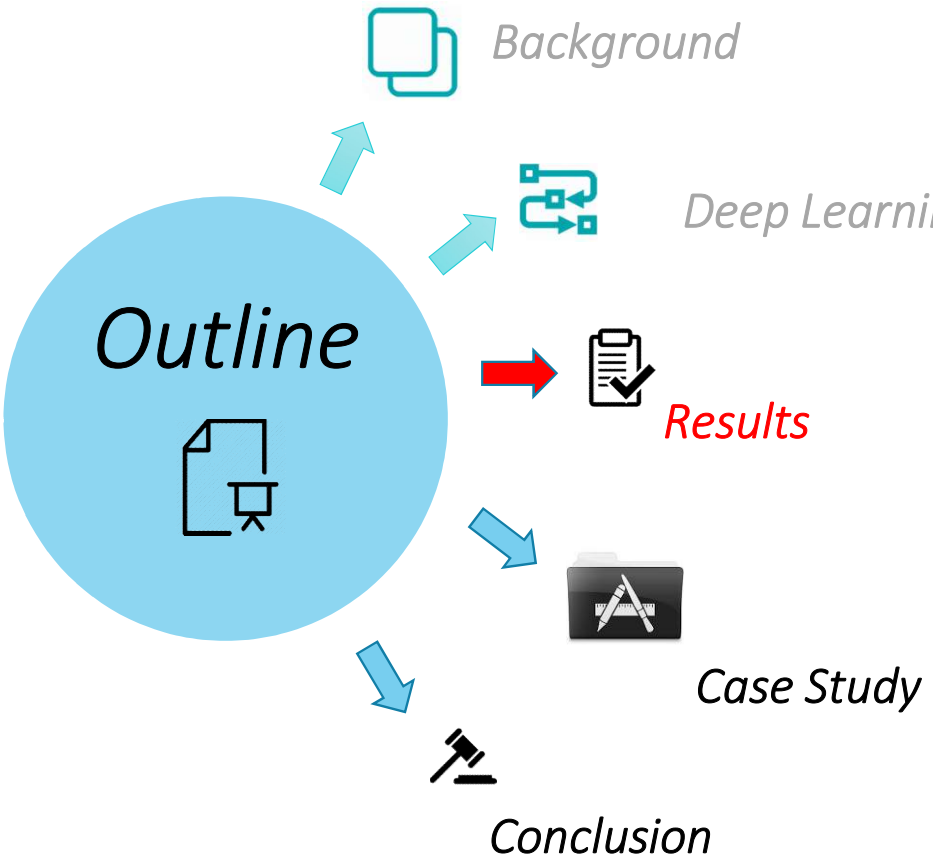
Hyper-Parameters				Training Time
Optimizer	**Learning Rate**	**Batch Size**	**Epochs**	**(minutes)**
ADAM	0.0001	64	25	115.84

DLRoute: Underfitting/Overfitting



The curves clearly show that the model is neither under-fitting nor over-fitting the data

Outline for Reminder of Talk



DLRoute: Performance Results

Overall Performance

Accuracy	Precision	Sensitivity	Specificity	M	Train Time	Test Time
97.4%	0.961	0.980	0.970	0.876	115.8 (min)	7.8 (ms)

Performance on Each Placement Phase

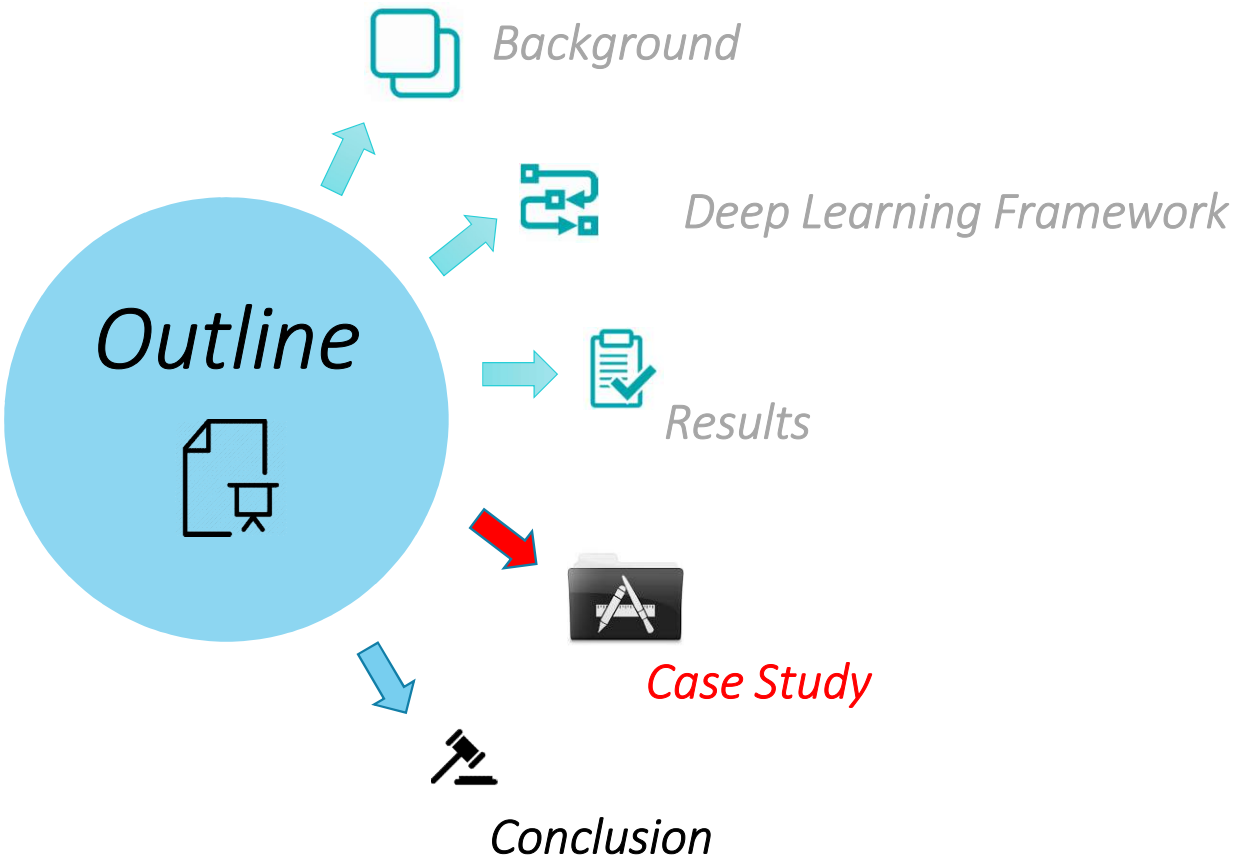
Phase	Accuracy	Precision	Sensitivity	Specificity	M
Global (Wirelength) Placement	0.988	0.955	0.993	0.986	0.967
Global(Congestion) Placement	0.958	0.944	0.962	0.954	0.915
Detailed Placement	0.983	0.987	0.995	0.826	0.864

DLRoute: Performance Results

Performance on Each Benchmark

Set	Accuracy	Precision	Sensitivity	Specificity	M
FPGA1	0.962	0.968	0.986	0.864	0.876
FPGA2	0.975	0.980	0.990	0.907	0.913
FPGA3	0.988	0.987	0.996	0.970	0.971
FPGA4	0.976	0.964	0.988	0.966	0.953
FPGA5	0.990	0.947	0.991	0.990	0.963
FPGA6	0.987	0.974	0.992	0.983	0.972
FPGA7	0.899	0.813	0.888	0.903	0.774
FPGA8	0.987	0.985	0.992	0.979	0.973
FPGA9	0.991	0.965	0.993	0.991	0.973
FPGA10	0.969	0.913	0.990	0.960	0.929
FPGA11	0.988	0.981	0.981	0.991	0.972
FPGA12	0.979	0.992	0.909	0.998	0.937

Outline for Reminder of Talk



DLRoute Case Study #1: Saving Router Time

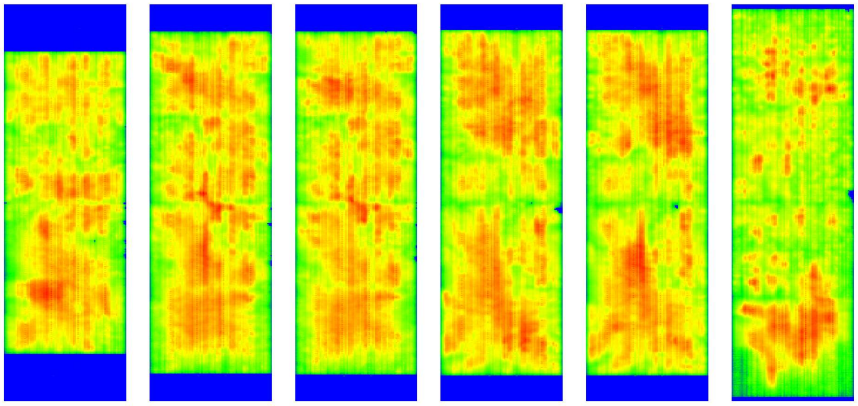
- The proposed routability predictor can be used to avoid costly, and futile place-and-route iterations
- The savings in time ranges from 42.7% to 82.1%

Placer	Routability		CPU Time			Saving
	Routable	Non-Routable	Routed	Unrouted	Total	
UTPlace[8]	317 (85%)	55 (15%)	315473	308239	623712	49.4%
Ripple[9]	336 (90%)	36 (10%)	338626	253180	591806	42.7%
Vivado2015.4	262 (70%)	110 (30%)	209402	964381	1173783	82.1%
Vivado2018.1	327 (88%)	45 (12%)	527227	402704	929931	43.4%

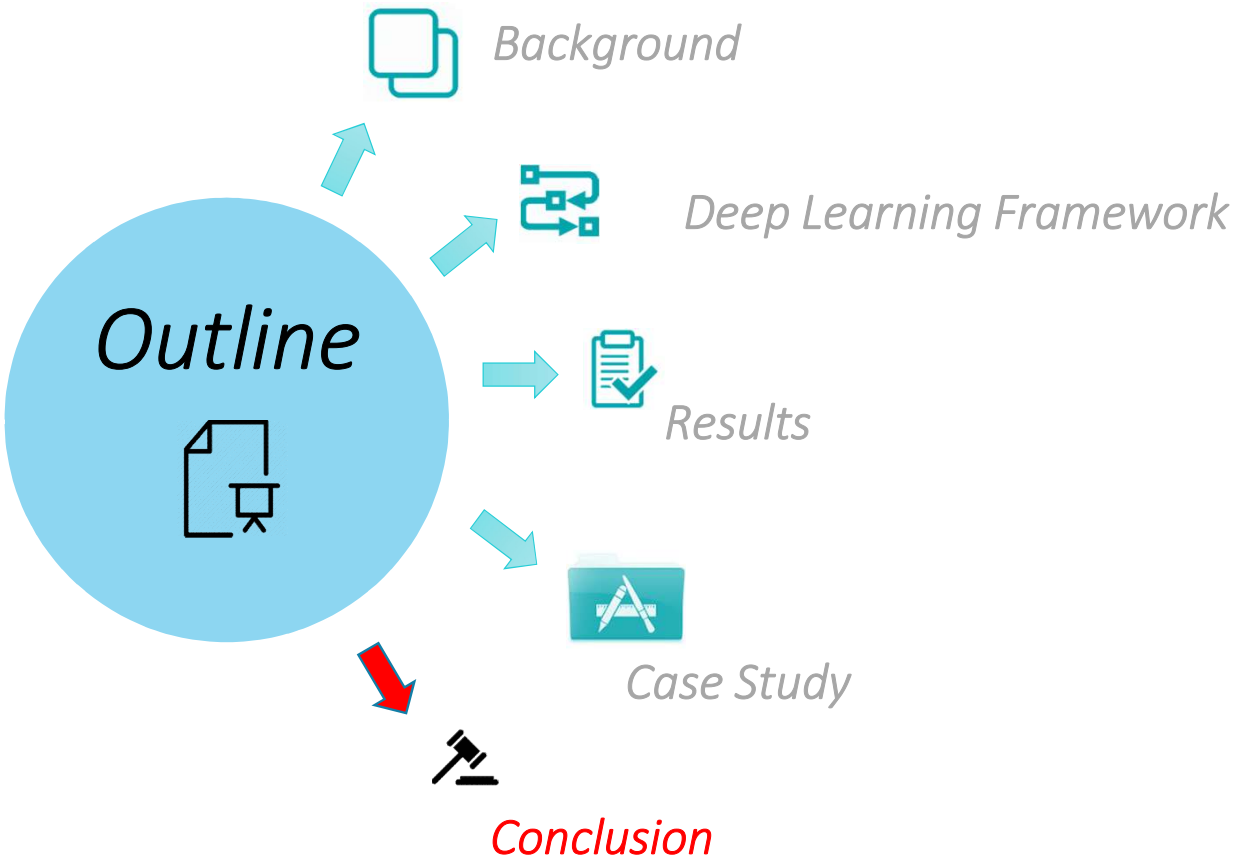
DLRoute Case Study #2: Feedback to the Placer

- Integrating the CNN into a placement tool can be used to adaptively improve its optimization strategy.
- To be able to route highly-congested placement, a feedback from the CNN is used to control the cell inflation parameters.
- The six highly-congested benchmarks are now routable.

Benchmark	Routing Results	
	Wirelength	CPU Time (seconds)
FPGA5-6	9900742	2639
FPGA5-11	11814937	6634
FPGA5-16	11858397	5497
FPGA5-19	12069961	4897
FPGA5-26	12035954	3235
FPGA7-7	9540692	3095



Outline for Reminder of Talk



Conclusions & Future Work

- A novel deep learning model for predicting FPGA routability during placement was proposed:
  - DLRoute can be used at any stage during the placement
  - DLRoute is capable of efficiently and accurately predicting the routability
  - DLRoute achieves on average a 97% accuracy to predict the routability of a produced placement
  - DLRoute can applied within any placement tool and it is architecture agnostic
- Our future work will focus on applying deep learning to further improve FPGA timing estimation and integrating it with the proposed routability model.

Thank you!

Guelph FPGA CAD Group
Website: https://fpga.socs.uoguelph.ca
Email: aalhyari@uoguelph.ca

References

[1] Xilinx. [n.d.]. UltraScale Architecture Configurable Logic Block User Guide. Retrieved from: http://www.xilinx.com/ support/documentation/user_guides/ug574-ultrascale-clb.pdf.

[2] Z. Abuowaimer, D. Maarouf, T. Martin, J. Foxcroft, G. Grewal, S. Areibi, and A. Vannelli, "GPLace3.0: Routability Driven Analytic Placer for UltraScale FPGA Architectures," *ACM Transaction on Design Automation of Electronic Systems*, vol. 23, no. 5, pp. 66:1–66:3, August 2018.

[3] Xilinx. [n.d.]. ISPD 2016 Routability-Driven FPGA Placement Contest. Retrieved from: http://www.ispd.cc/contests/ 16/ispd2016_contest.html.

[4] https://www.xilinx.com/products/silicon-devices/fpga.html

[5] D. Maarouf, A. Alhyari, Z. Abuowaimer, T. Martin, A. Gunter, G. Grewal, S. Areibi, and A. Vannelli, 2018, August. Machine-learning based congestion estimation for modern fpgas. In *2018 28th International Conference on Field Programmable Logic and Applications (FPL)* (pp. 427-4277). IEEE.

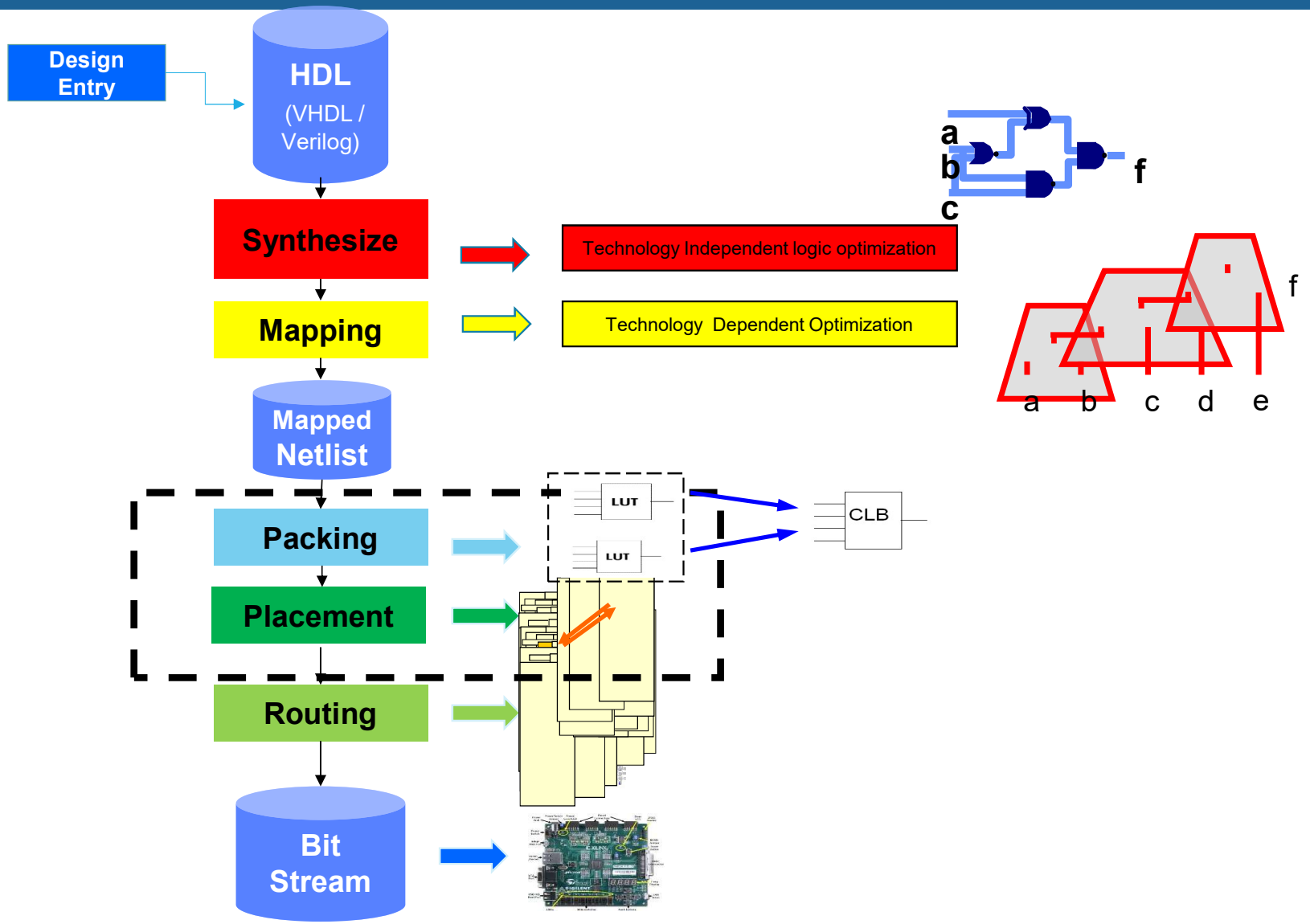
Supplementary Slides

Why Use Machine/Deep Learning for EDA?

Machine Learning has unique features:

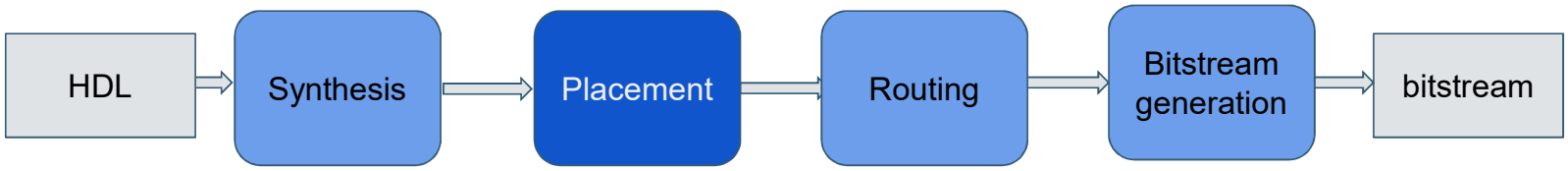
- **Data-driven:** ML can learn from data to recognize complex patterns, insights and relationships in data
- **No explicit programming:** ML has the ability to extract knowledge and draw inferences from data
- **May assist in cutting CPU time:** ML can replace time consuming steps in FPGA CAD flow. ML can efficiently and accurately replace congestion estimation and routability prediction tools.
- **Provides guidance to the flow:** ML is able to provide an informative feedback that can be used by an adaptive placement flow to enhance its performance and reduce CPU time

Traditional FPGA CAD Flow



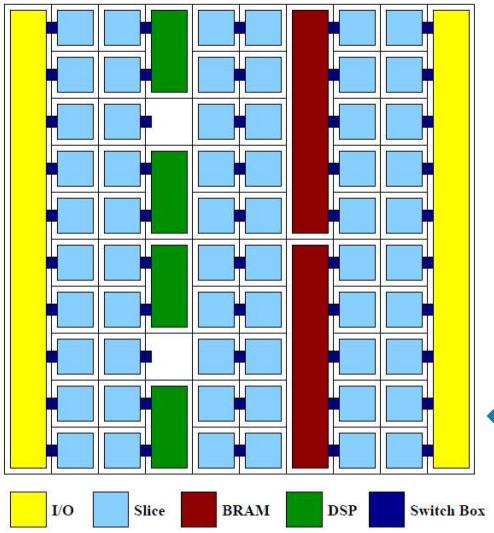
The FPGA CAD Flow

- Hardware design done by modelling system in HDL
- Synthesis: netlist generated
- Placement: Components placed on chip
- Routing: Connecting signals routed
- Bitstream generated to program FPGA

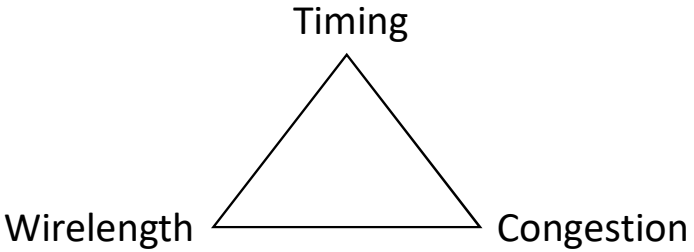


FPGA Placement: Challenges

- There are multiple resources constraints:
  - Heterogeneity: (LUT, FF, DSP, BRAM)
  - LUT sharing constraints
  - FF control-set constraints



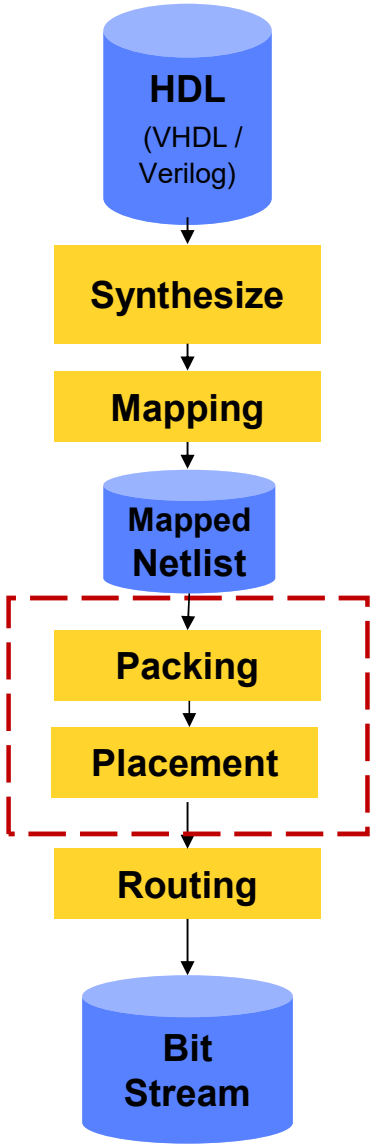
- Multiple conflicting objectives:
  - Wirelength, Timing, Congestion, etc..

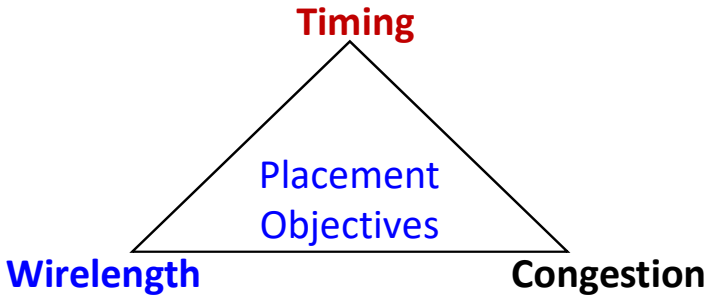


- High compile time:
  - Designs complexity
  - Millions of cells (logic blocks)
  - Runtime takes more than a day

Need to introduce Intelligence in FPGA Placement

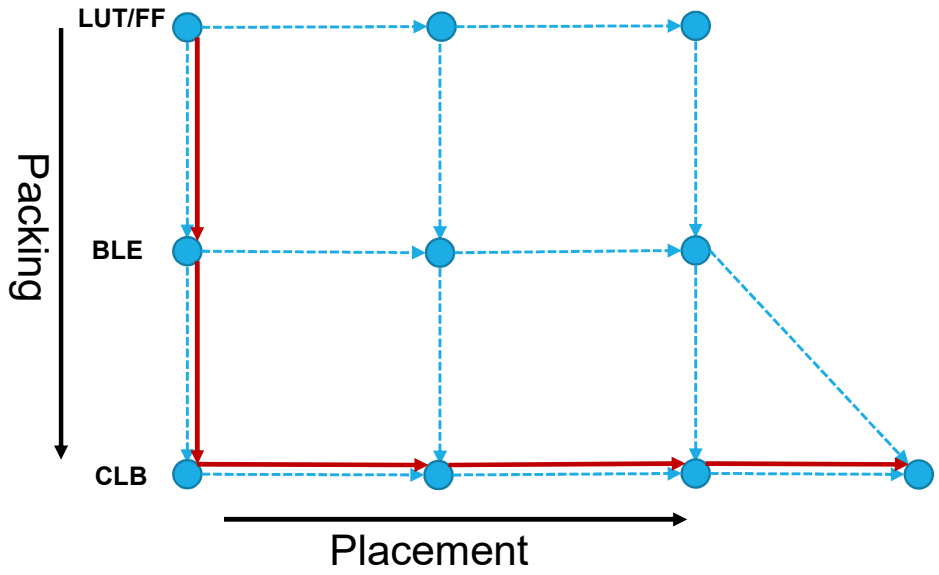
Packing and Placement





Properly managing the *interdependence* between packing and placement is **key to** optimizing wirelength, timing, and congestion!

Complete Packing: LUT/FF → BLE → CLB



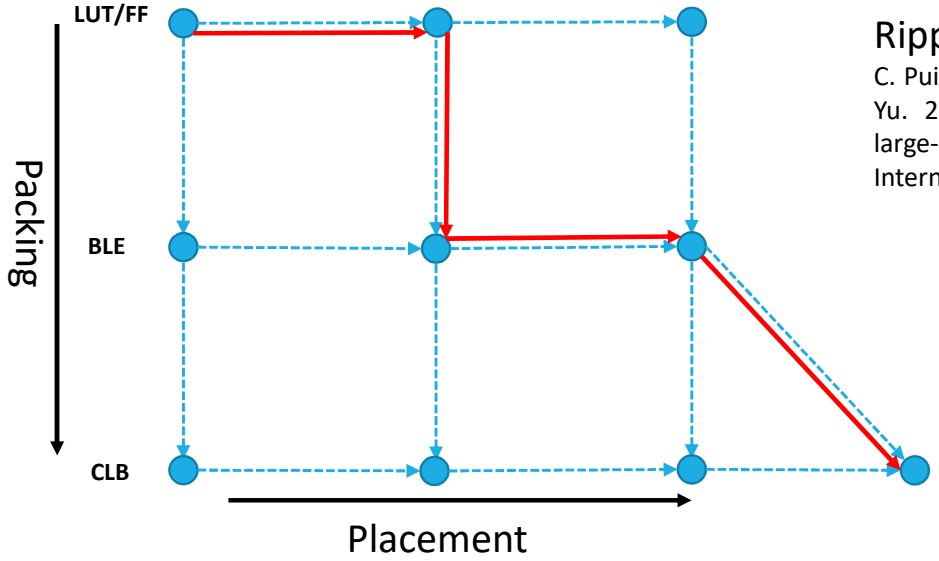
VPR

V. Betz and J. Rose, "VPR: A new packing, placement and routing tool for FPGA research," Field Programmable Logic and Applications, 1997, pp. 213-212.

Traditional Pack-Place-Legalize:

- This technique **tends to pack** LUTs and FFs **at an early stage** of the optimization thus may be difficult to unpack CLBs at a later stage if congestion is encountered thus **may lead to unroutable solutions!!**.

Partial Packing: LUT/FF → BLE



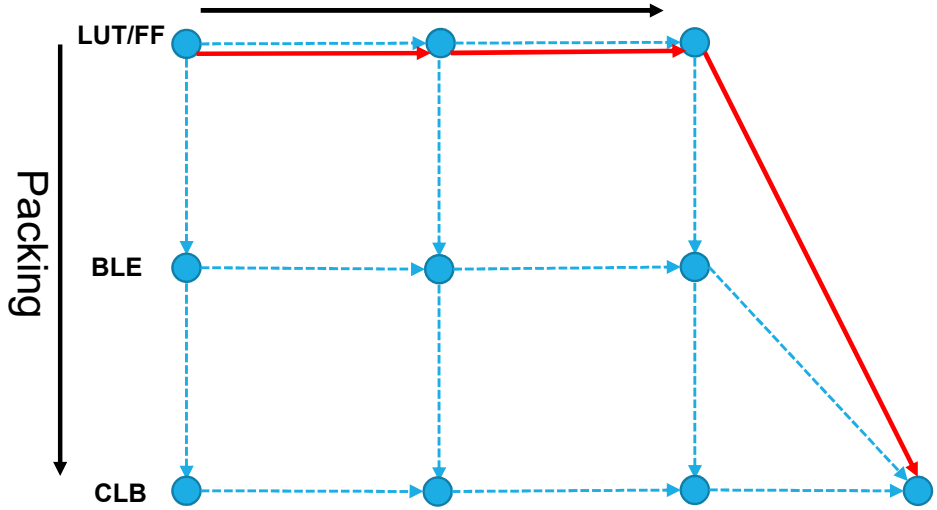
RippleFPGA (2nd Place in ISPD16):

C. Pui, G. Chen, W. Chow, K. Lam, P. Tu, H. Zhang, E. Young, and B. Yu. 2016. RippleFPGA: A routability-driven 883 placement for large-scale heterogeneous FPGAs. In Proceedings of the International Conference on Computer-Aided 884 Design. 1–8.

Place-SemiPack-Place-Legalize:

- More flexible than complete packing.
- However, Semi Packing **tends to produce sub optimal solutions** due **to congestion encountered** later in the placement stage.

No Packing (Flat Placement)

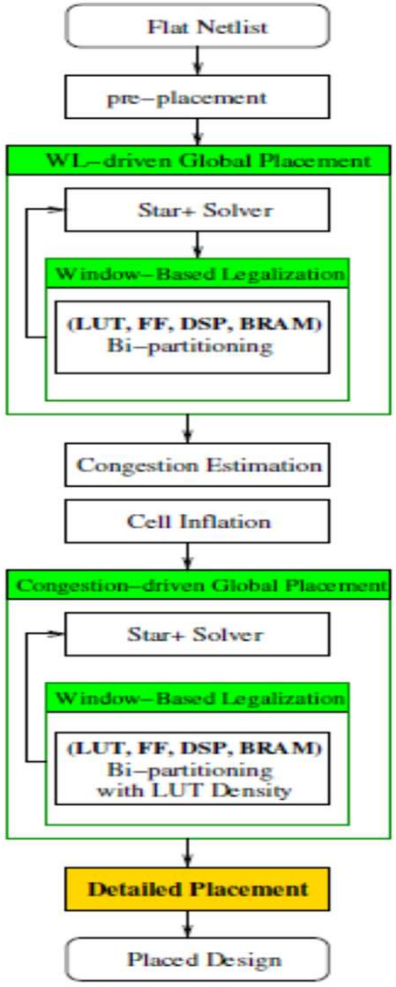


Place-Legalize:

- **Flat placement** allows LUTs and FFs to move throughout the placement flow **thus unrestricting** the solution space.
- However, may be slow!

GPlace3.0

Z. Abuowaimer, D. Maarouf, T. Martin, J. Foxcroft, G. Grewal, S. Areibi and A. Vannelli. GPlace3.0: Routability-Driven Analytic Placer for UltraScale Architectures, ACM Transactions on Design Automation of Electronics Systems, Volume 23, Issue 5, 2018, pp. 1-33.



b) GPlace 3.0

Analytical Placement

- Prior approaches to placement use simulated annealing.
- Recently, more attention has been directed towards analytic placement, which scales better on large problem instances

- Analytic placement approach

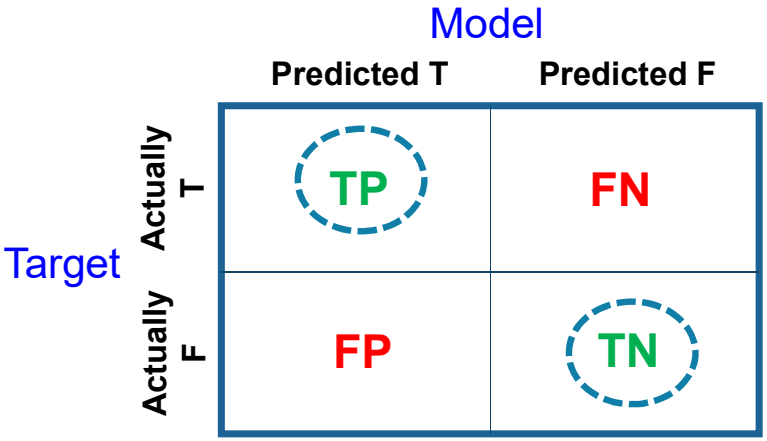
```
1: Convert netlist to graph using Net model
2: Perform pin propagation
3: repeat
4:    solve non-linear equation system
5:    partition solution to enforce legality
      constraints
6: until termination criteria satisfied
```

Experimental Setup

- GPlace was implemented **using C**, compiled using gcc (Red Hat 4.4.7-18) compiler.
- Binary executable files were provided from other teams for Ripple and UTplaceF placers
- Experiments were run on **an Intel** (Xeon CPU E3-1270 v5 @ 2.6 GHz) **processor** with 16 GB RAM.
- Placement solutions were routed using Xilinx **Vivado 2015.4**, with a **patch** applied to make Vivado compatible with the modified Bookshelf Format used by both academic placement tools.
- The **Scikit Learn** machine learning library for the Python programming language was used to implement the various classification models.
- **Keras** and **Tensorflow** are used to develop the deep learning frameworks.

Binary Classification Evaluation Metrics 1/2

- A confusion matrix is an N x N matrix, where N is the number of target labels (classes)
- It shows the number of correct and incorrect predictions made by the classifier compared to the actual outcomes (target labels) in the actual data
- E.g., binary classification problem (e.g., two classes 0|1 or T|F)



Accuracy: the proportion of the total number of predictions that are correct

TN + TP / (TN + TP +FP + FN)

Binary Classification Evaluation Metrics 2/2

1. Accuracy: $T_N + T_P / (T_N + T_P + F_P + F_N)$
2. Recall (Sensitivity): $T_P/(T_P + F_N)$
3. Precision: $T_P/(T_P + F_P)$
4. Specificity: $T_N/(T_N + F_P)$
5. F1-Score: 2 x (Precision x Recall)/(Precision + Recall)
6. $M = (T_N * T_P - F_P * F_N) / \sqrt{(T_P + F_P)(T_P + F_N)(T_N + F_P)(T_N + F_N)}$

Data Size Comparison for MLRoute & DLRoute

	MLRoute	DLRoute
Total amount of data (train + test)	856	26551

- The data that was used to train, validate, and test DLRoute is 31x more the data of MLRoute

The Presented Frameworks

Area Targeting in Placement	Presented Frameworks
Congestion Estimation	MLCong, DLCong
Routability Prediction	MLRoute, DLRoute
Flow Selection	MLSelect

Congestion Management

Congestion Management (Inflation followed by Spreading):

1. Identify switches that are highly congested
2. Inflate all cells (LUTS) that belong to this switch by a certain amount
3. Use a bipartitioning Legalization technique to spread and move LUTs to neighboring regions to relieve current switch from overflow and congestion

GPlace3.0: LUT Inflation

Algorithm 1 LUT Inflation

Require: $Cong_B$ is the average congestion for the top 10% most congested switches
Require: $congThresh$ is the congestion threshold
Require: $lutCount$ is the number of LUTs

1: $S \leftarrow C_1 + C_2 \cdot max(congThresh, Cong_B)$
2: $lutSum \leftarrow 0$
3: **for all** LUTs **do**
4: $\quad lutSum \leftarrow lutSum + inputs(LUT) + outputs(LUT)$
5: **end for**
6: $\mu_{LUT} \leftarrow \frac{lutSum}{lutCount}$
7: **for all** LUTs **do**
8: $\quad cong \leftarrow switchCongestion(LUT)$
9: $\quad$ **if** $cong \geq 0.5$ **and** $cong < 0.675$ **then**
10: $\quad\quad ratio \leftarrow 0.8$
11: $\quad$ **else if** $cong \geq 0.675$ **and** $cong < 0.85$ **then**
12: $\quad\quad ratio \leftarrow 0.7$
13: $\quad$ **else if** $cong \geq 0.85$ **and** $cong < 1.025$ **then**
14: $\quad\quad ratio \leftarrow 0.6$
15: $\quad$ **else if** $cong \geq 1.025$ **and** $cong < 1.2$ **then**
16: $\quad\quad ratio \leftarrow 0.5$
17: $\quad$ **else if** $cong > 1.2$ **then**
18: $\quad\quad ratio \leftarrow 0.4$
19: $\quad$ **end if**
20: $\quad size[LUT] = 1 + S \cdot \left(\frac{inputs(LUT)+outputs(LUT)}{ratio \cdot \mu_{LUT}} - 1\right)$
21: **end for**

Parameter Name	Description	Default Value
lower	Lower bound of range of congestion values to inflate.	0.5
upper	Upper bound of range of congestion values to inflate.	1.2
adap6 parameter 1 (C_1)	See Algorithm 1 line 1.	0.36
adap6 parameter 2 (C_2)	See Algorithm 1 line 1.	0.28
congestion threshold	Congestion value above which a switch is considered congested.	0.65

GPlace3.0 : LUT Inlation

$$\text{density(LUT)} = 1 + S.\left(\frac{\text{inputs(LUT)} + \text{outputs(LUT)}}{\text{ratio}.\mu_{LUT}} - 1\right)$$

Machine Learning in Electronic Design Automation (EDA)

- Three problems in FPGA placement flow that are targeted using machine learning and deep learning:
  - Congestion estimation
  - Routability prediction
  - Flow selection