BUFFORMER: A GENERATIVE ML FRAMEWORK FOR SCALABLE BUFFERING

RONGJIAN LIANG*, SIDDHARTHA NATH*, ANAND RAJARAM*, JIANG HU+, AND MARK REN*

*NVIDIA +TEXAS A&M UNIVERSITY

OUTLINE

- $\boxed{\text{INTRODUCTION}}$
- PROBLEM FORMULATION
- METHODOLOGY
- EXPERIMENTAL VALIDATION
- CONCLUSTIONS AND FUTURE DIRECTIONS

WHAT'S BUFFERING & HOW GENERATIVE ML HELPS?

Buffering: insert buffers to interconnects to improve timing, *NP-complete* problem

BufFormer: the first generative ML-based framework for scalable buffering

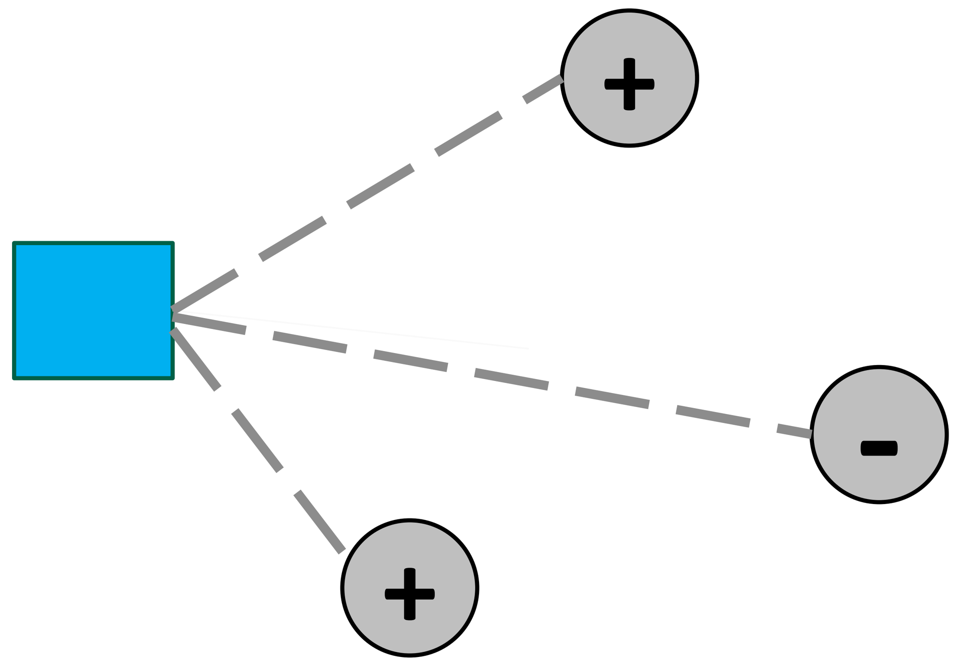
OUTLINE

- INTRODUCTION
- PROBLEM FORMULATION
- METHODOLOGY
- EXPERIMENTAL VALIDATION
- CONCLUSTIONS AND FUTURE DIRECTIONS

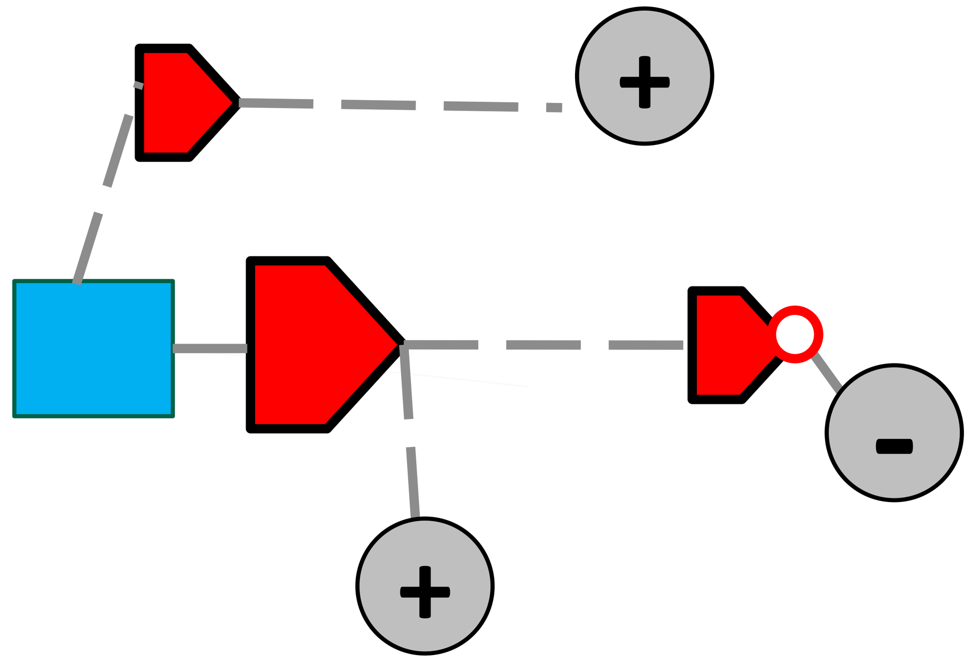
PROBLEM FORMULATION

Sinks, their polarities
and locations

Driver, its
input slew
and location

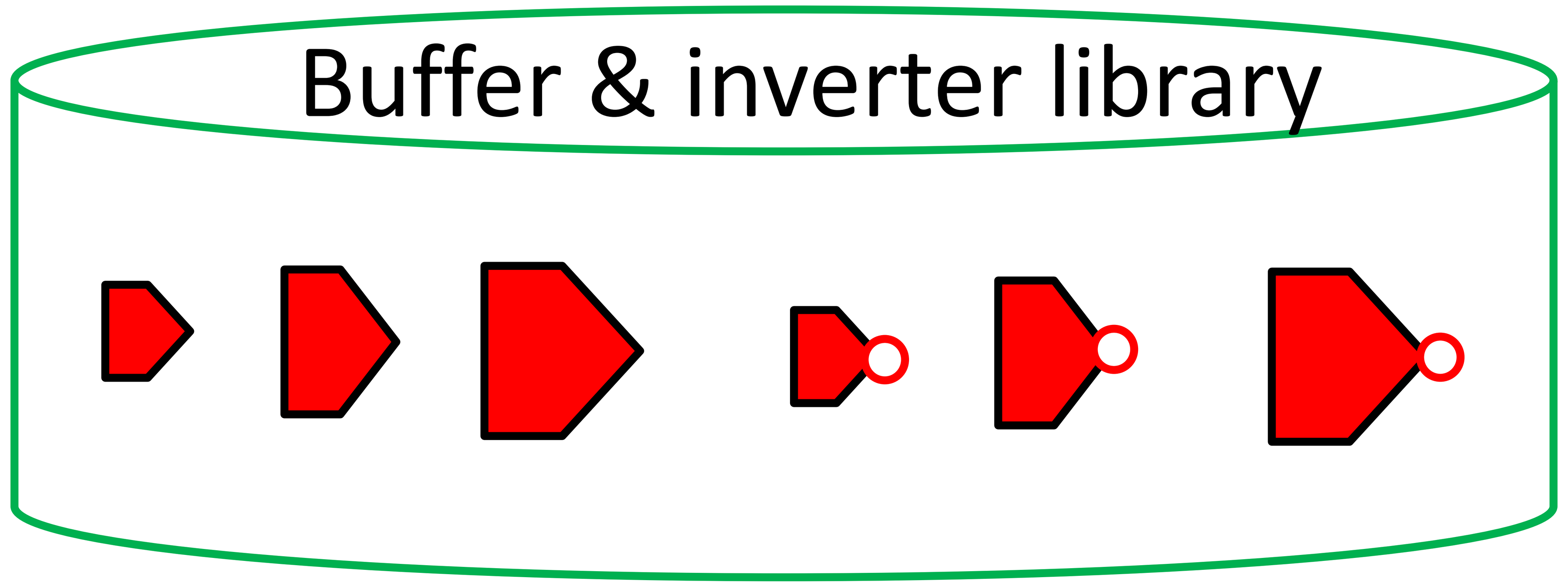


Delay targets for
each driver-sink pair



Buffer-embedded tree

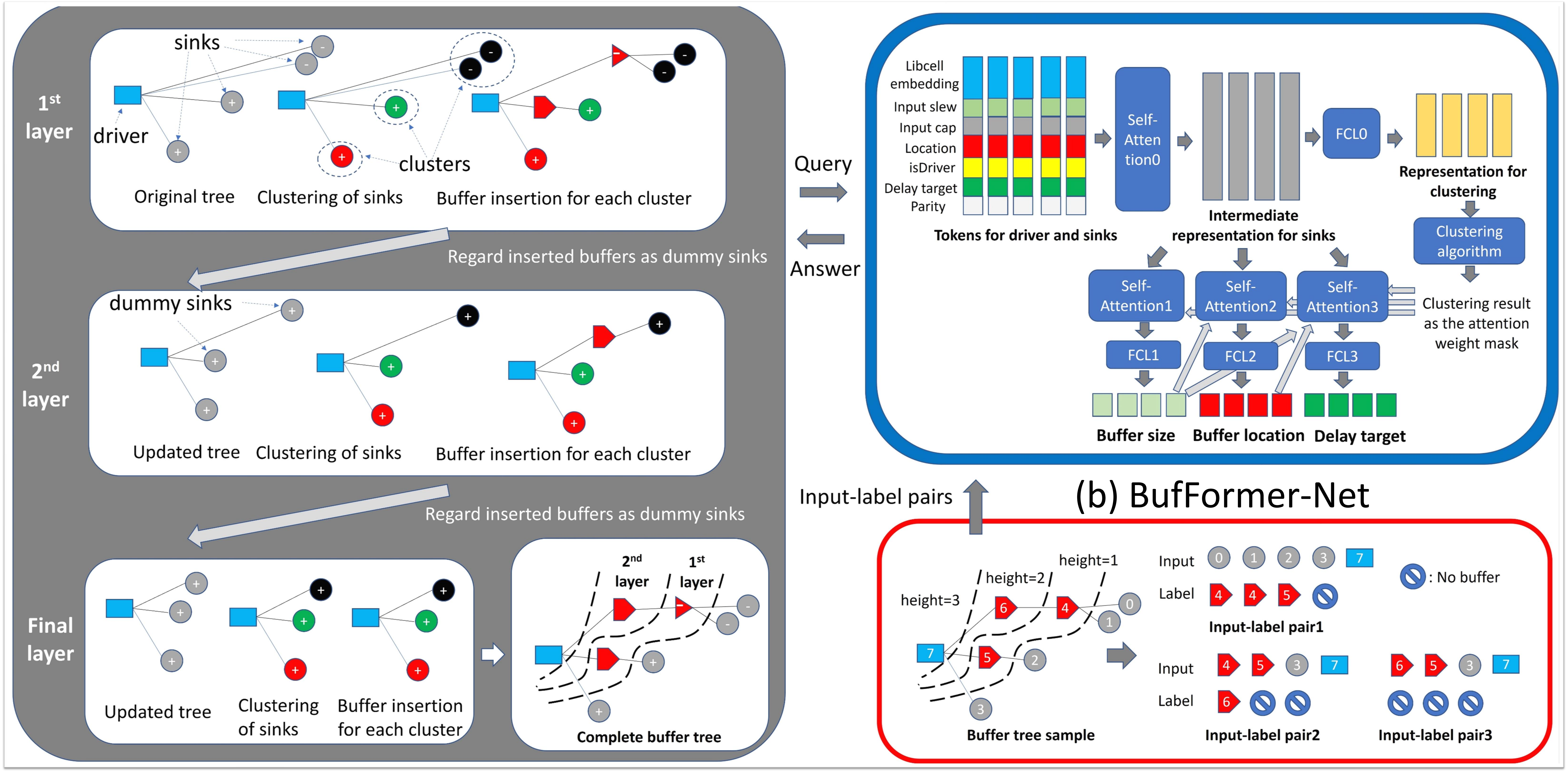
- Abstract tree topology
- Location&size of buffers



OUTLINE

- INTRODUCTION
- PROBLEM FORMULATION
- METHODOLOGY
- EXPERIMENTAL VALIDATION
- CONCLUSTIONS AND FUTURE DIRECTIONS

BUFFORMER FRAMEWORK



(b) BufFormer-Net

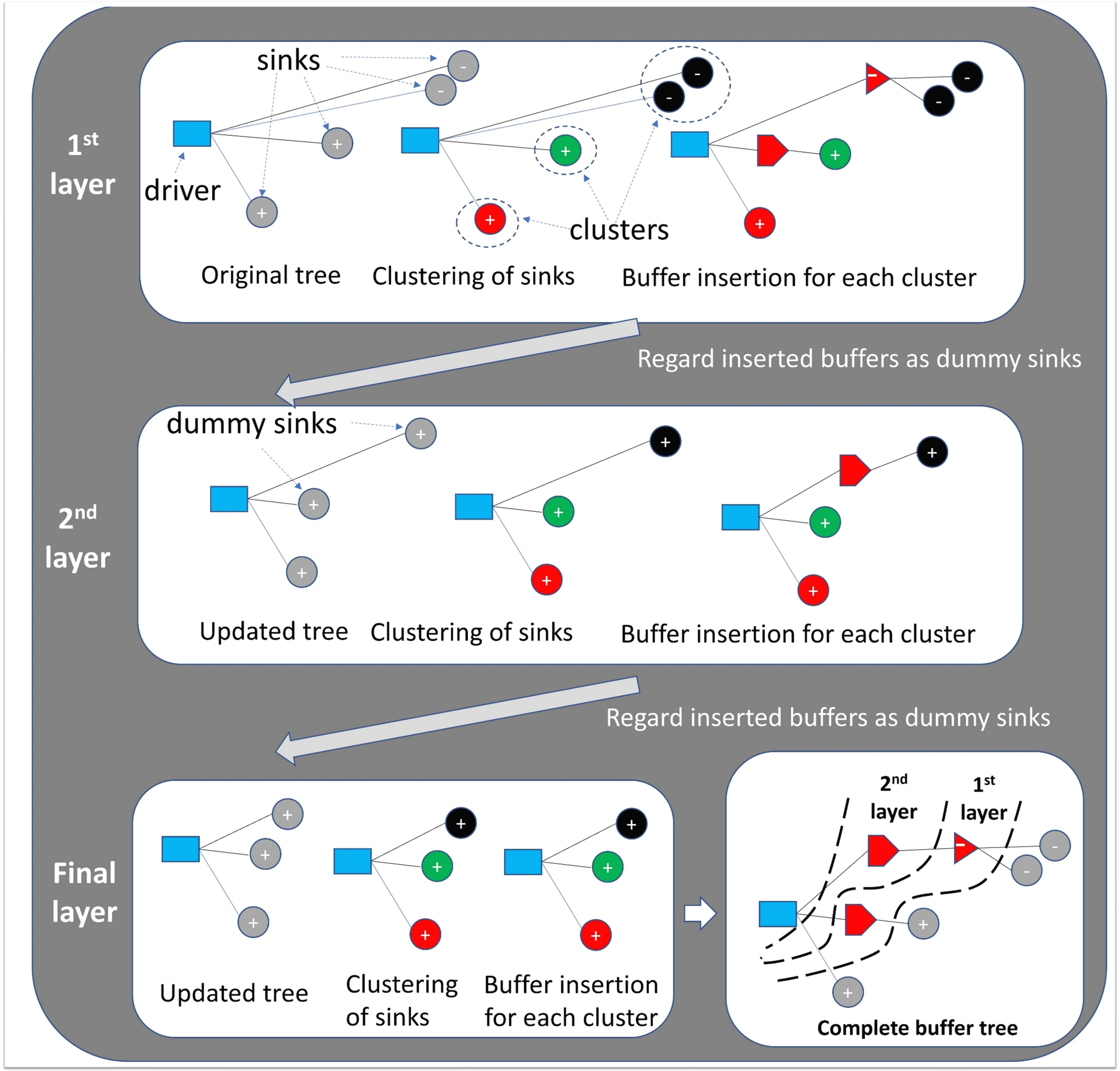
(a) Tree generation process

(c) Training scheme

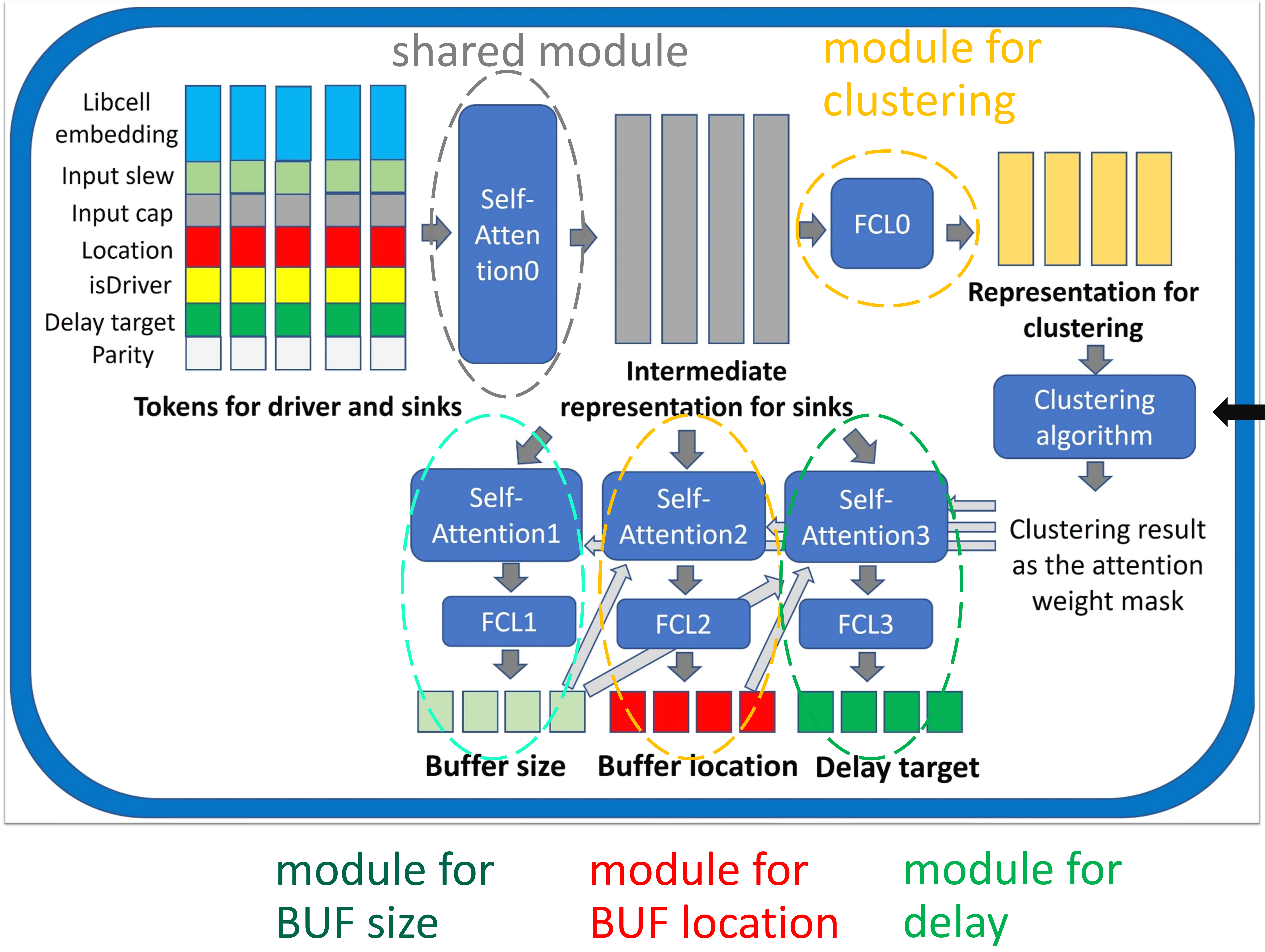
TREE GENERATION PROCESS

Properties

- Recursive process
- Layer-by-layer bottom-up process
- Clustering-base process



BUFFOMER-NET

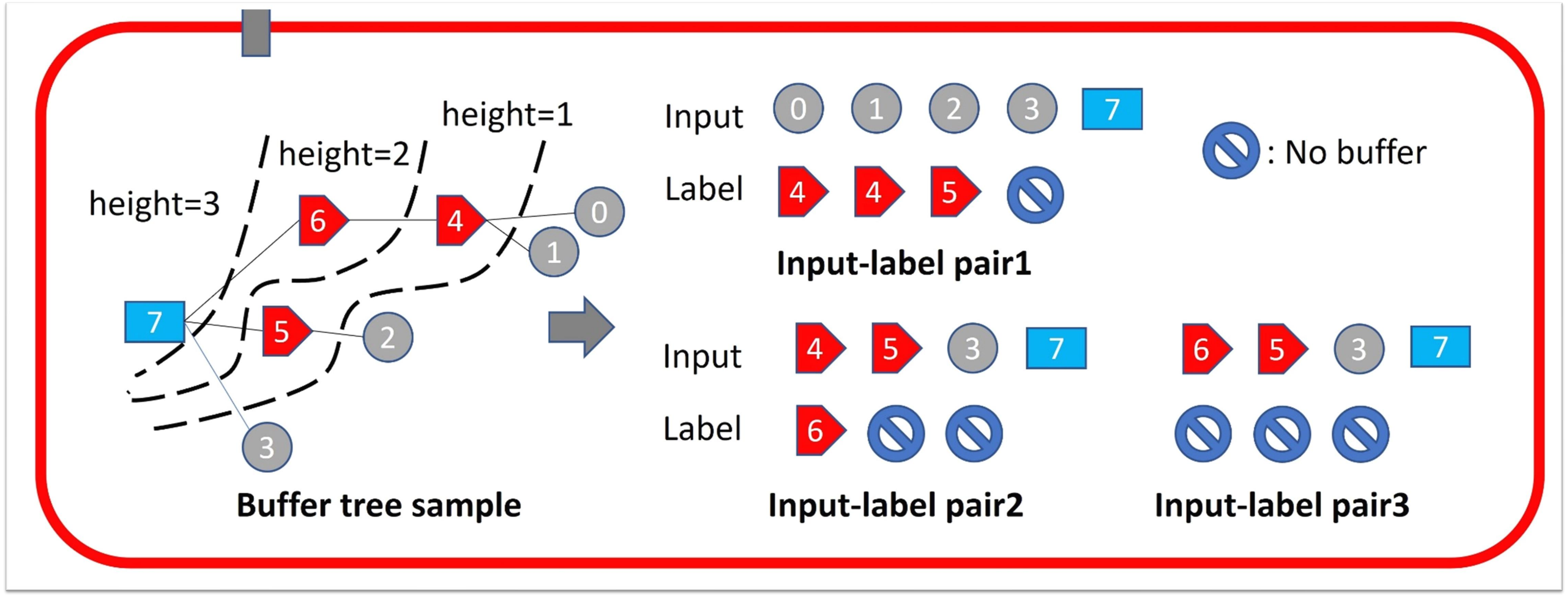


Connection Clustering

1. If two sinks are close in the representation space, then regard them as connected
2. Each connected component of sinks is viewed as a cluster

SELF-SUPERVISED TRAINING SCHEME

Input-label preparation:



Loss functions:

1. Contractive loss for clustering
2. Focal loss for BUF size
3. MSE loss for BUF location
4. MSE loss for delay

Multi-objective training for shared parameters:

1. Compute the gradients of each loss w.r.t. shared parameters
2. Compute a minimum-norm vector in the convex hull of the set of gradient vectors
3. Update parameters in the direction of the minimum-norm vector

OUTLINE

- INTRODUCTION
- PROBLEM FORMULATION
- METHODOLOGY
- EXPERIMENTAL VALIDATION
- CONCLUSTIONS AND FUTURE DIRECTIONS

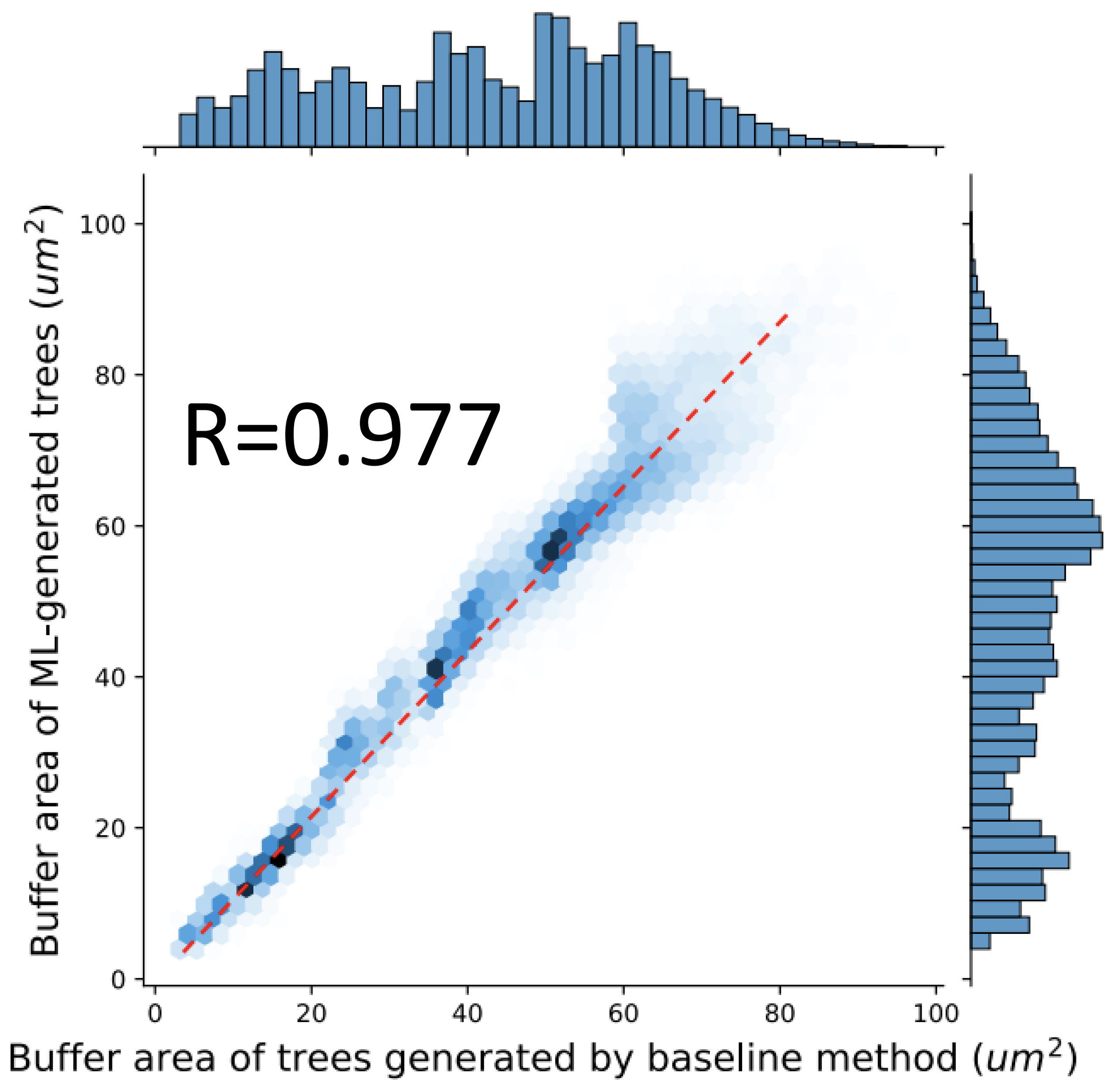
EXPERIMENT RESULTS

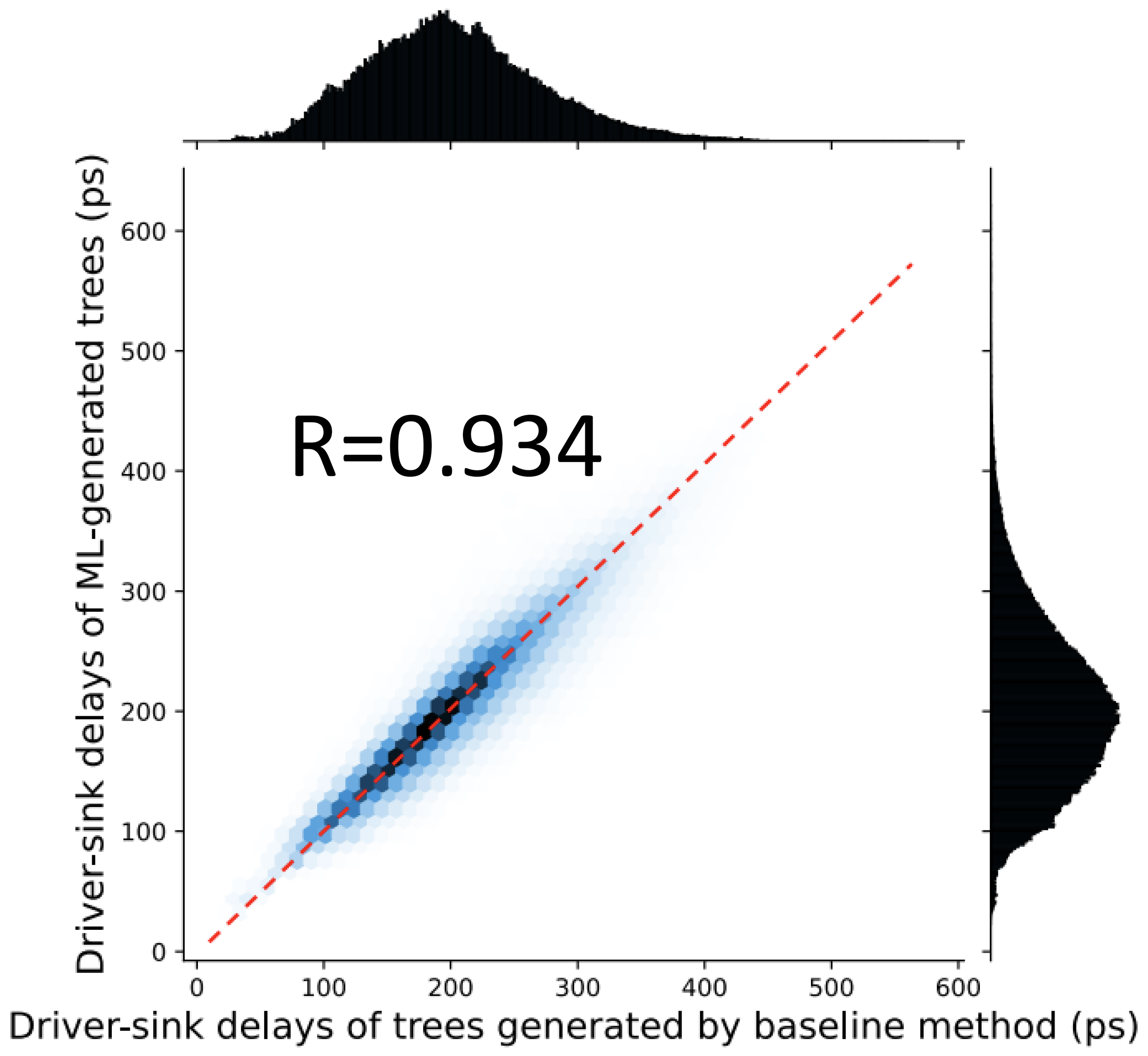
Setups

- Baseline: FLUTE + Lillis implemented in OpenPhySyn
- NanGate45 cell library
- 5 buffers & 4 inverters
- Artificial nets

Table 1: Characteristics of Samples

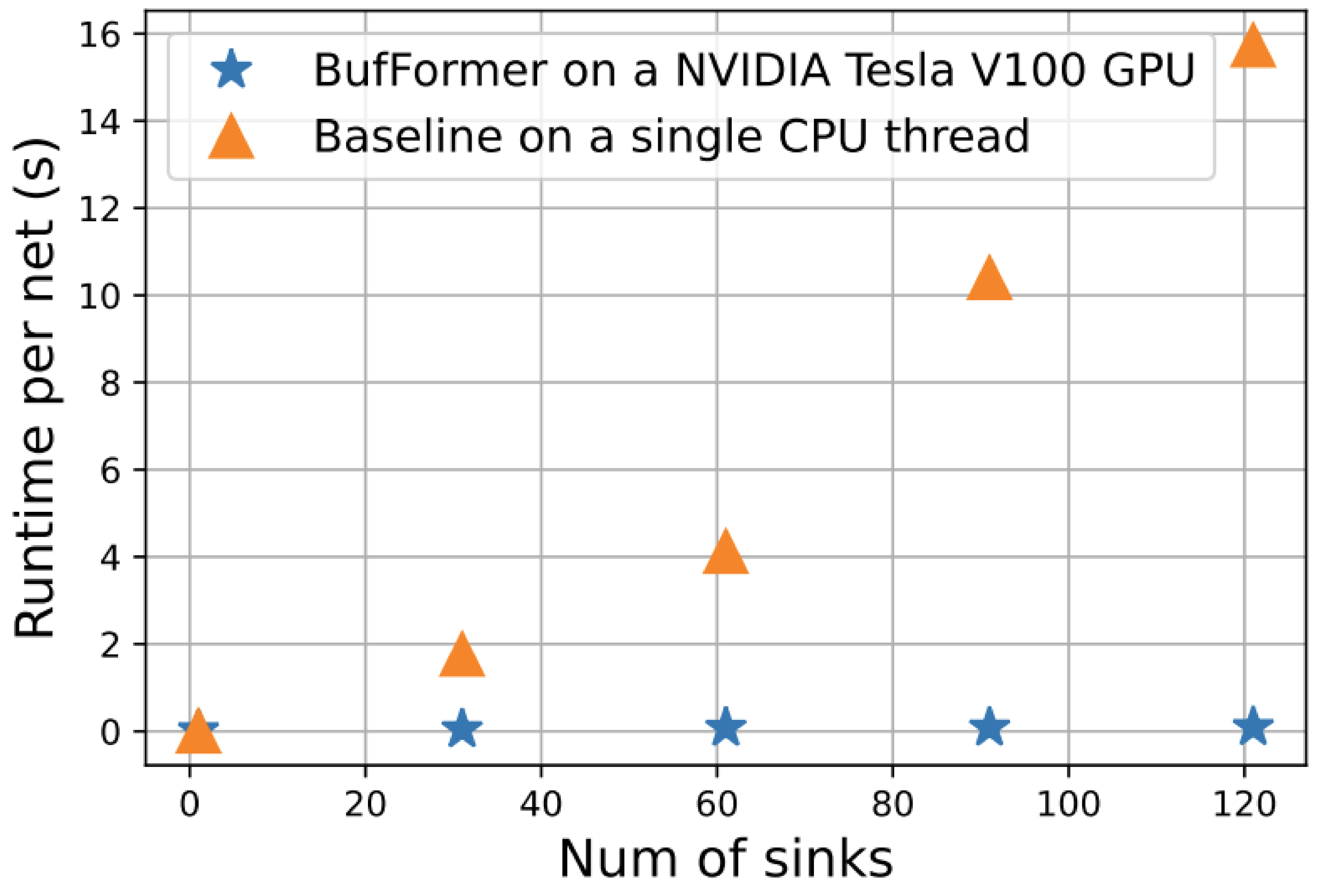
characteristic	train set	test set
net count	23083	1620
tree count	343004	32031
sink count range & avg.	[1,150], 48	[0,98], 56
buffer area range & avg. (μm^2)	[0,151], 37	[0,99], 43
driver-sink delay range & avg. (ps)	[0,1128], 186	[0,609], 200
HPWL range & avg. (μm)	[0, 2214] , 863	[0, 2796], 1270





Results:

- Highly comparable performance to the baseline
- Up to *160X speedup* for large nets
- Parity constraint, capacitance and slew limits are met

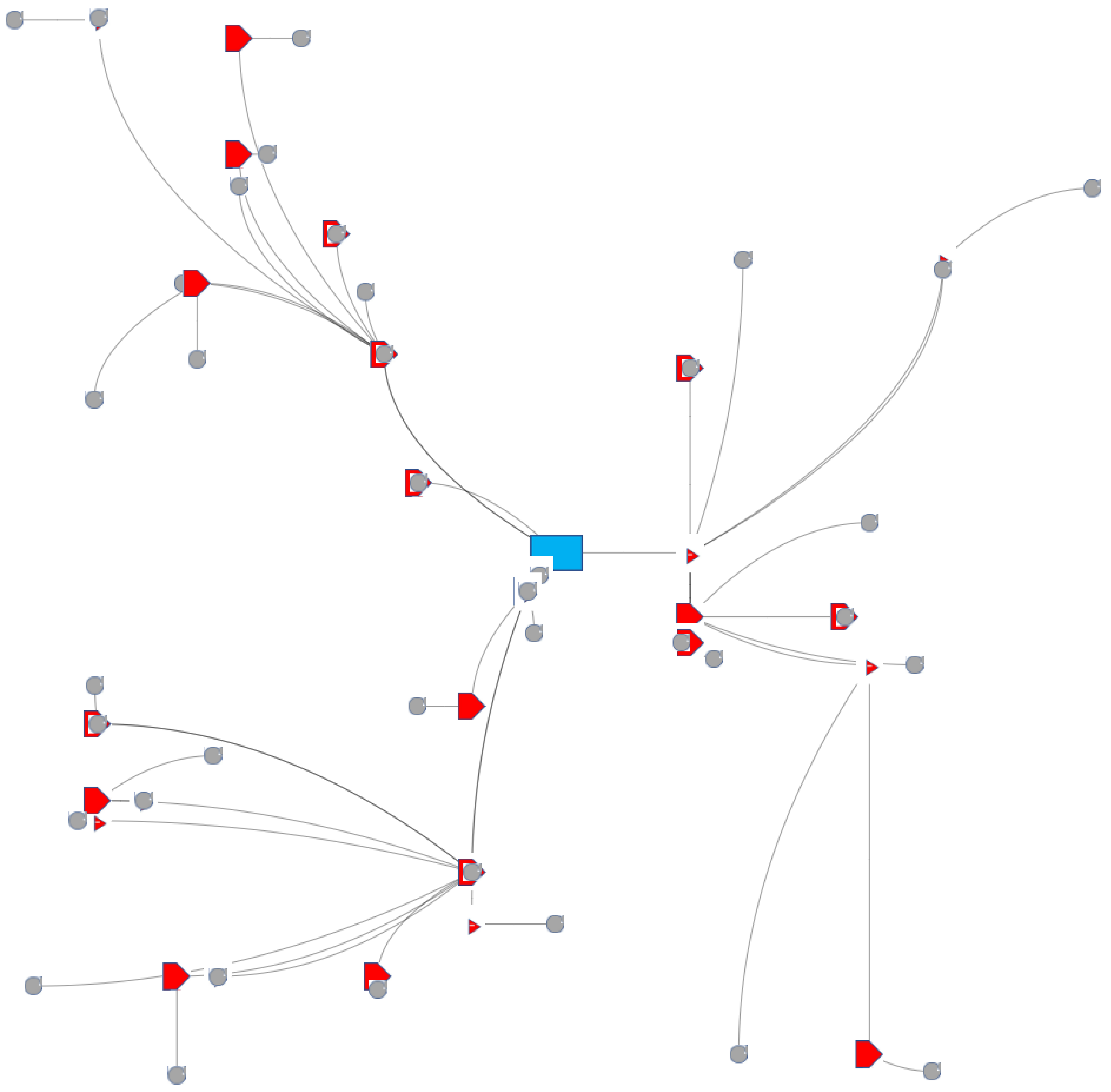


EXPERIMENT RESULTS (CONT.)

Table 2: Results of Ablation Studies

factors	method	cluster acc	libcell acc	loc RMSE (μm)	delay tar. RMSE (ps)	buf. area diff. (μm^2)			driver-sink delay diff.(ps)		
						cor	mean	std	cor	mean	std
	default	**92.6%**	**91.6%**	**34**	**10**	**0.977**	**4.8**	**4.9**	**0.934**	**0**	**25**
data amount	40%	91.1%	88.7%	36	11	0.964	4.6	5.9	0.762	7	51
	25%	87.0%	84.5%	40	14	0.939	1.3	7.1	0.761	8	50
model size	larger	93.1%	92.5%	34	10	0.975	6.3	5.3	0.931	0	25
	smaller	92.9%	90.7%	34	10	0.970	4.7	5.4	0.927	0	26
train loss	weighted	82.9%	95.8%	34	8	0.976	4.1	4.5	0.873	0	35
model arch.	separate	90.8%	95.5%	36	8	0.964	6.8	6.1	0.762	7	51
clustering algorithm	AC	/	/	/	/	0.974	4.2	4.9	0.916	0	28
	AP	/	/	/	/	0.972	6.1	5.4	0.905	0	30
	DBSCAN	/	/	/	/	0.932	-2.1	7.9	0.676	11	67

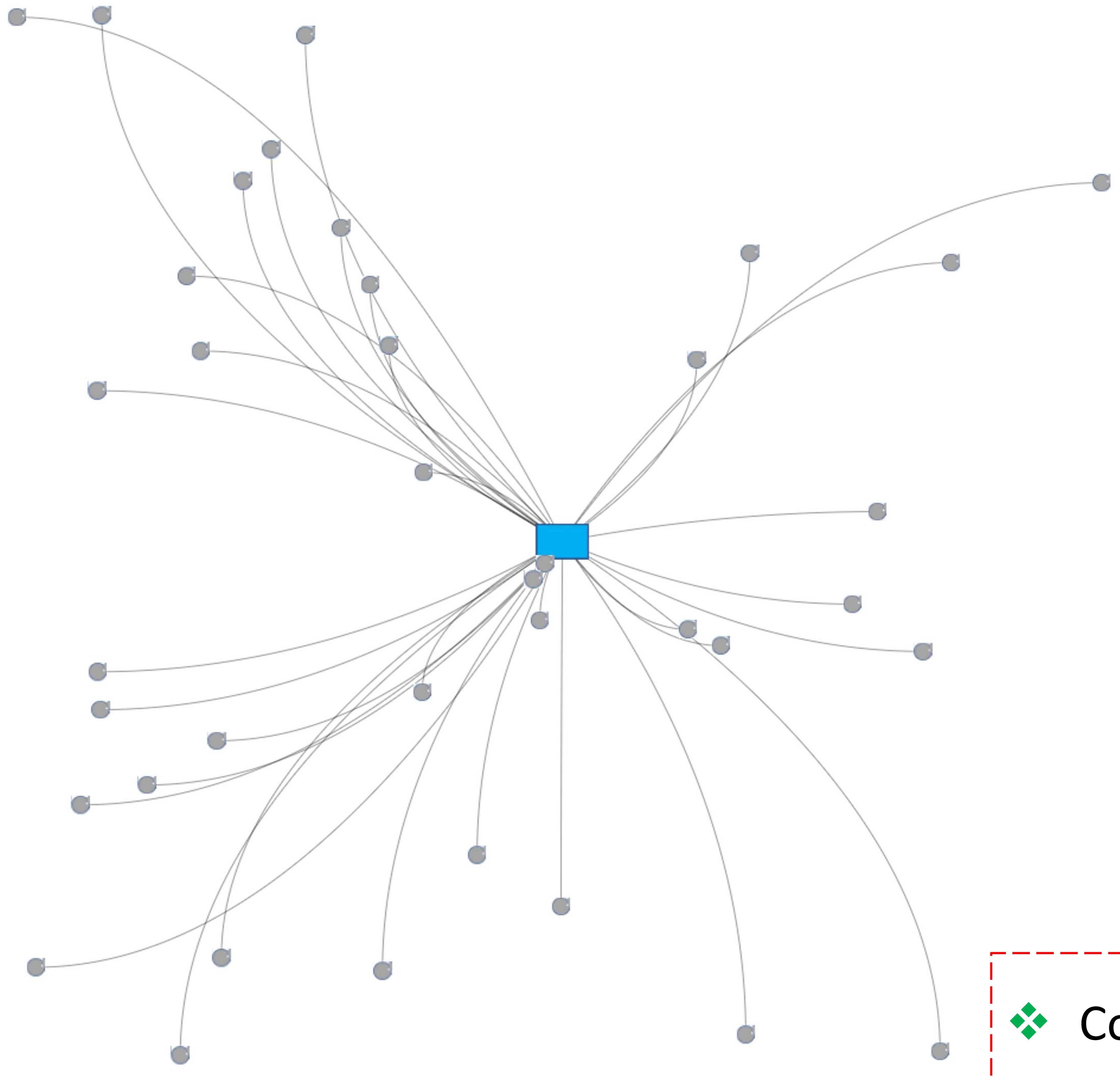
SNAPSHOT OF TREE GENERATION BY BUFFORMER



Tree generated by baseline (FLUTE+Lillis)

Total BF area: 31.7um2

Avg driver-sink delay: 155ps



Tree generated by our BufFormer

Total BUF area: 28.5um2

Avg driver-sink delay: 143ps

- Comparable performance
- Up to *160X speedup*
- Meet constraints

OUTLINE

- INTRODUCTION
- PROBLEM FORMULATION
- METHODOLOGY
- EXPERIMENTAL VALIDATION
- CONCLUSTIONS AND FUTURE DIRECTIONS

CONCLUSIONS

- The first successful attempt at generative ML-based buffering
- A generative ML framework, named Bufformer, which can learn from archived samples and generate buffered tree without Steiner tree construction
- Compared with a FLUTE-Lillis baseline running on a single CPU, BufFormer can generate buffered trees for unseen nets very close to baseline results, and achieve up to 160X speedup on a GPU

FUTURE DIRECTIONS

- Train BufFormer with commercial tool post-routing data
- Extend to handle realistic layout environment
- Circuit-level optimization