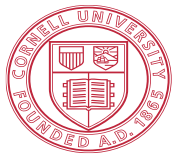
Accurate Operation Delay Prediction for FPGA HLS Using Graph Neural Networks

Ecenur Ustun*, Chenhui Deng*, Debjit Pal, Zhijing Li, Zhiru Zhang

Electrical and Computer Engineering, Cornell University

November 3, 2020



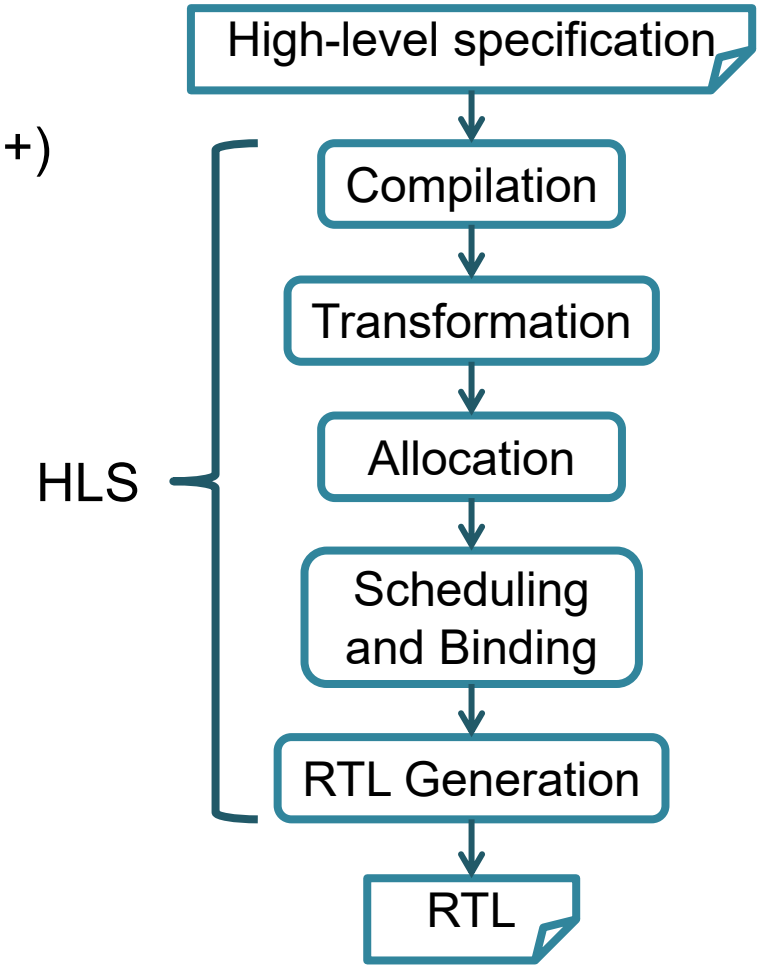
Cornell University

*Equal contributions

CSL

FPGA High-Level Synthesis (HLS)

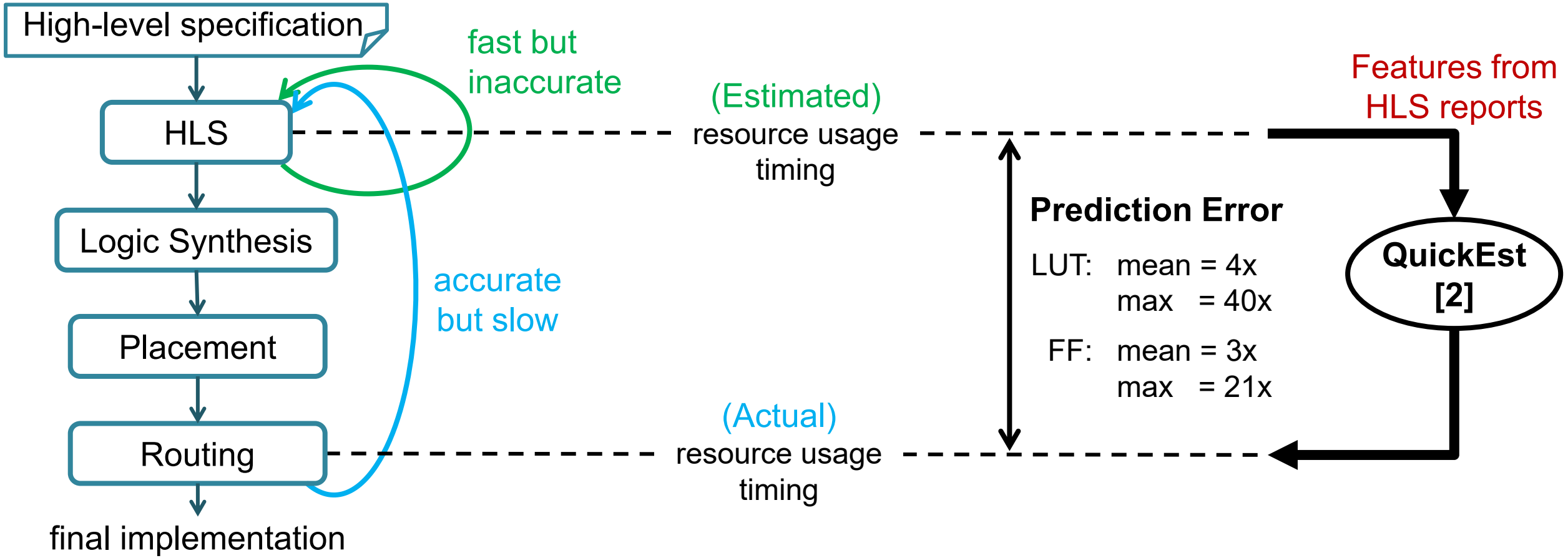
- Higher productivity in specialized hardware design
  - Specify hardware behavior in high-level languages (e.g., C/C++)
- Commercial HLS tools
  - Vivado HLS (Xilinx)
  - i++ (Intel)
  - Catapult-C (Mentor Graphics)
- Academic HLS tools
  - LegUp [1]
  - Bambu [2]



[1] A. Canis et al. LegUp: High-Level Synthesis for FPGA-based Processor/Accelerator Systems. FPGA, 2011.
[2] C. Pilato et al. Bambu: A Modular Framework for the High Level Synthesis of Memory-intensive Applications. FPL, 2013.

Delay Prediction in HLS

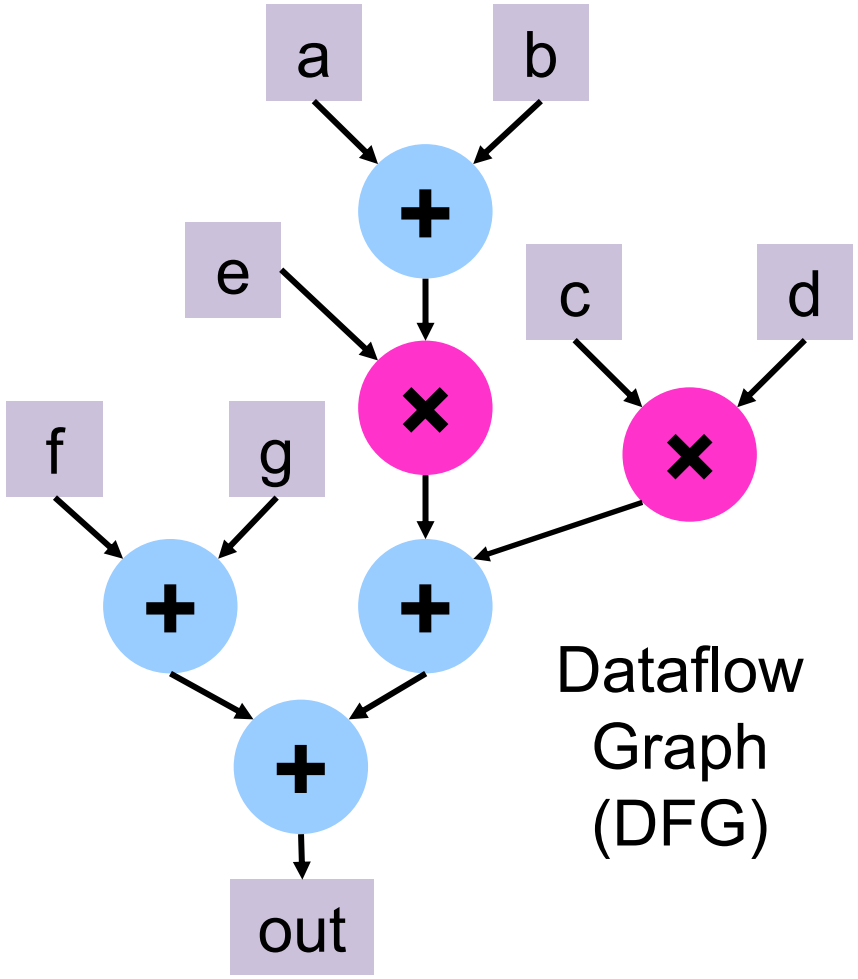
- Accurate delay prediction in earlier stages is crucial [1]



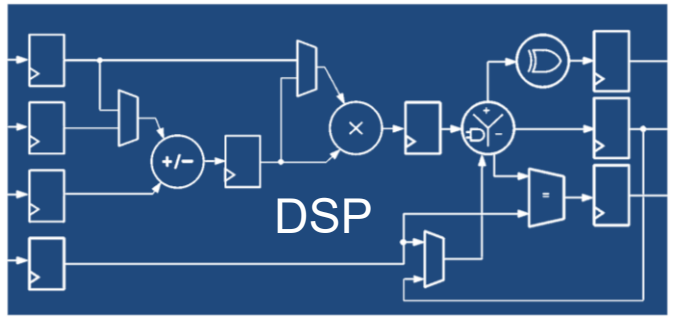
[1] R. Nane et al. A Survey and Evaluation of FPGA High-Level Synthesis Tools. TCAD, 2015.
[2] S. Dai et al. Fast and Accurate Estimation of Quality of Results in High-Level Synthesis with Machine Learning. FCCM, 2018.

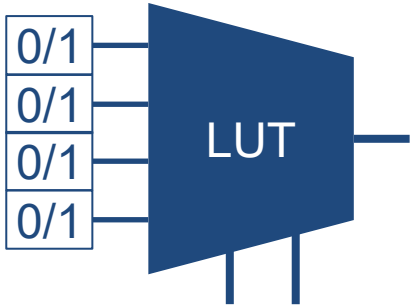
Learning Operation Mapping in HLS

- Introduce mapping-awareness in HLS [1]



Dataflow Graph (DFG)



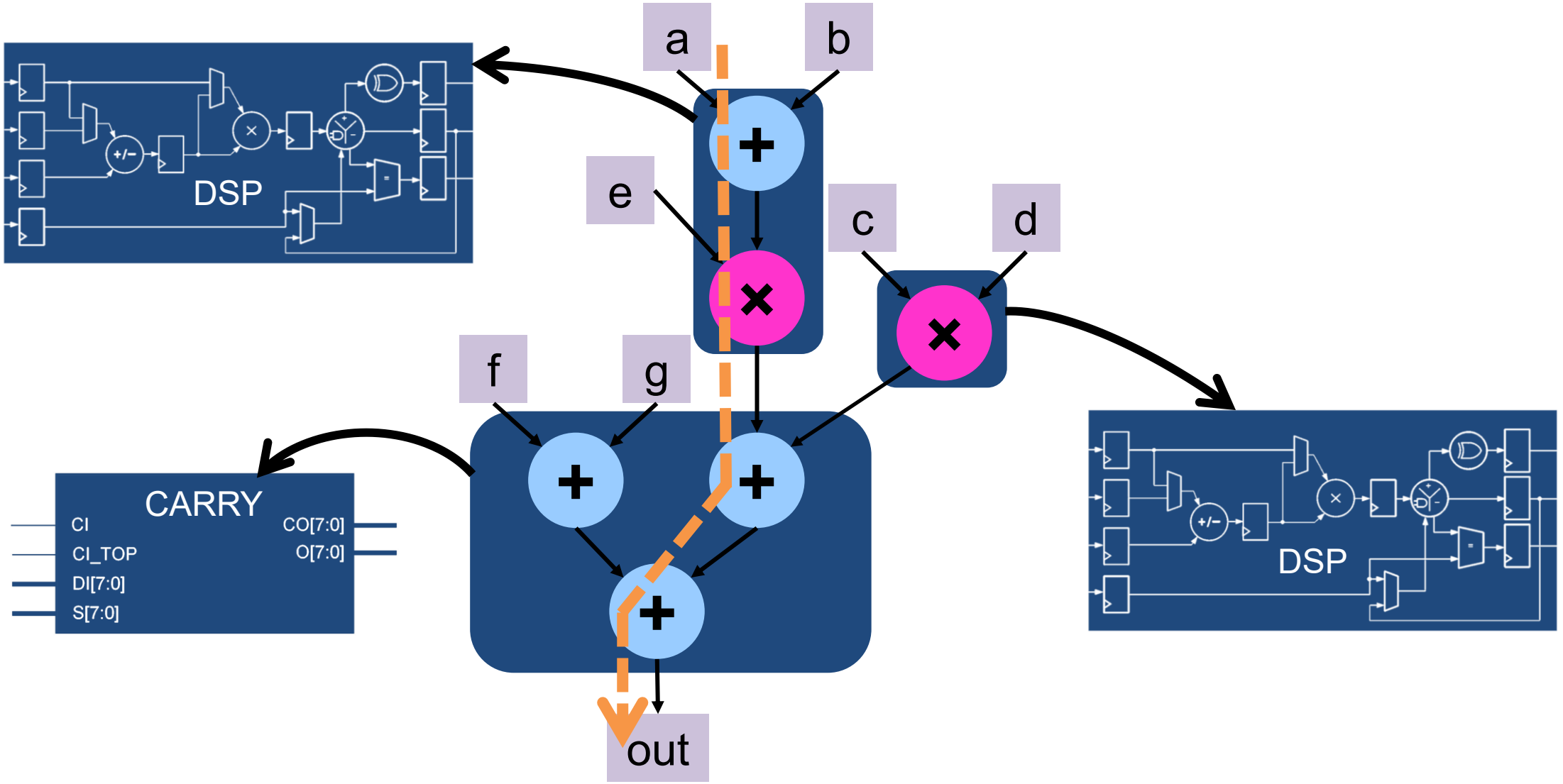


Can mapping patterns be learned?

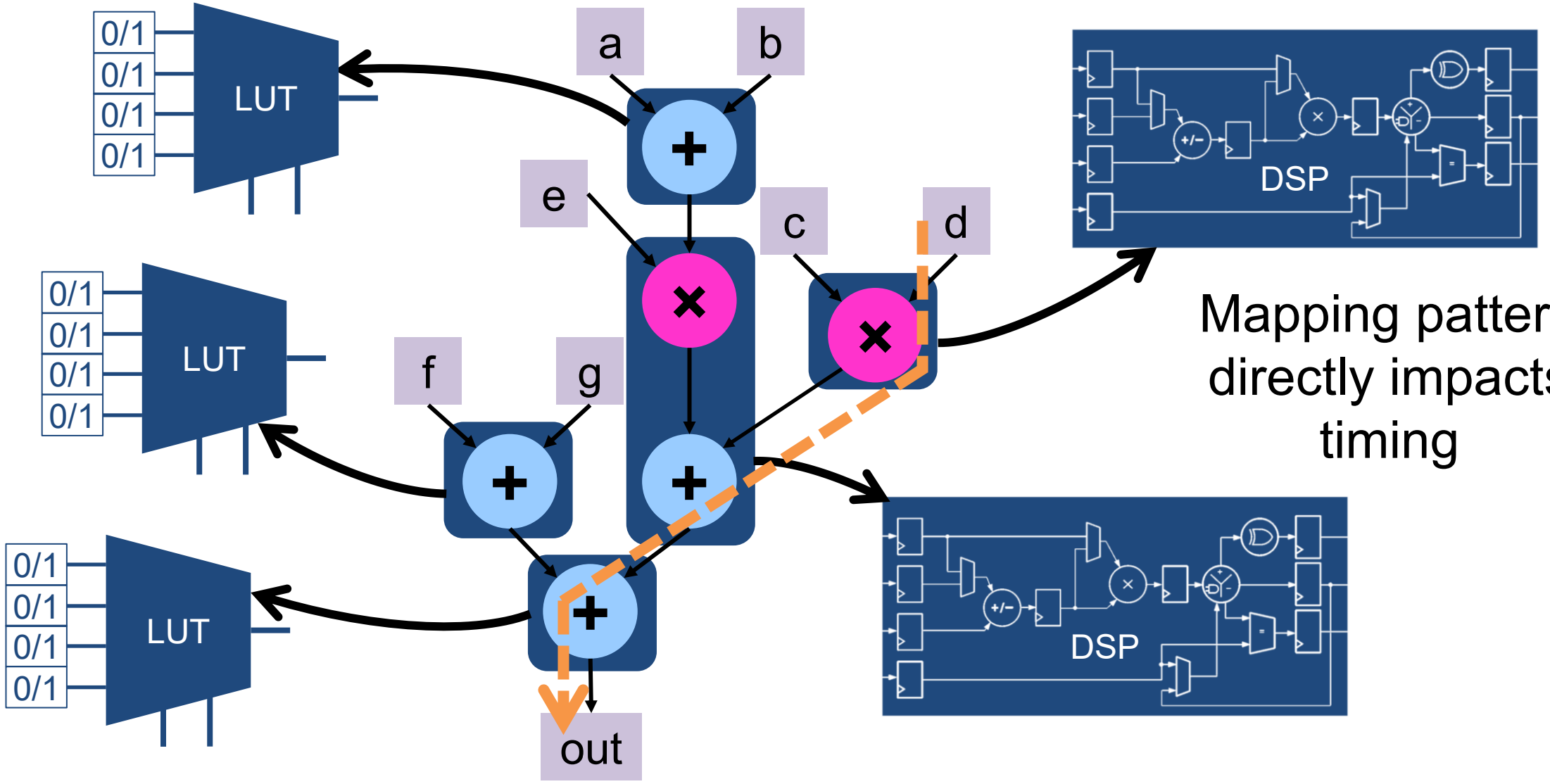
FPGA resources

[1] M. Tan et al. Mapping-Aware Constrained Scheduling for LUT-Based FPGAs. FPGA, 2015.

Learning Operation Mapping in HLS: Motivating Example



Learning Operation Mapping in HLS: Motivating Example

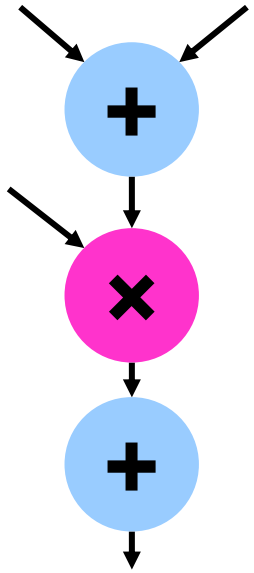


Our Approach

- Learn mapping of HLS operations onto FPGA device resources using graph neural networks
- Characterize delay in HLS based on learned mapping patterns
  - 72% improvement in HLS operation delay prediction

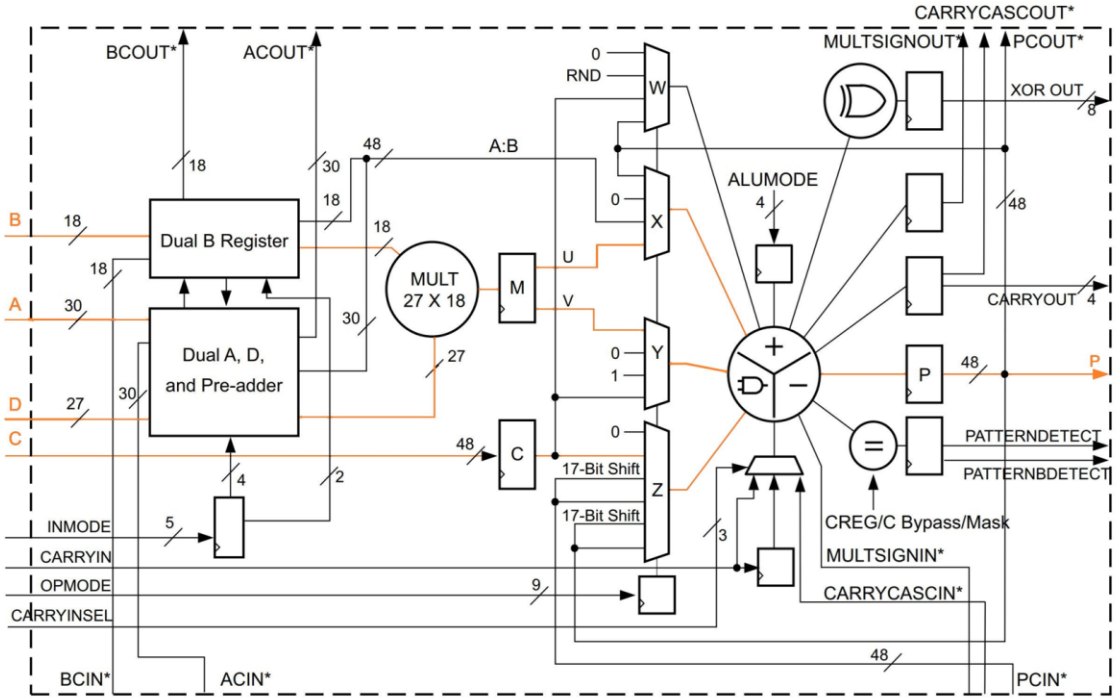
DSP Mapping

- DSPs are widely used for high-performance complex functions
- Matching HLS subgraphs with DSP blocks
  - Commercial HLS tools follow hard-coded rules to infer DSP mapping



subgraph

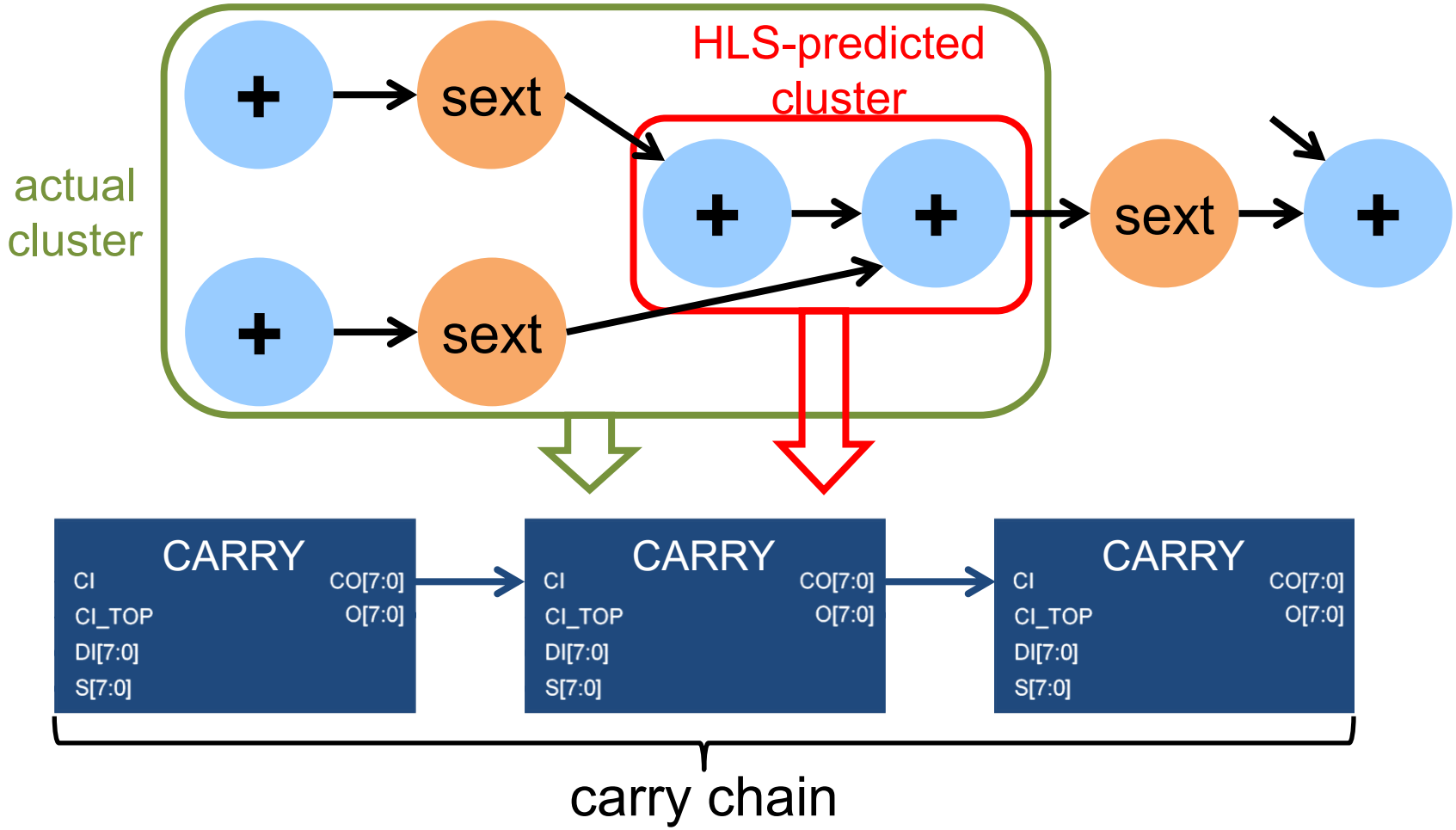
Goal: Automatically learn mapping patterns using machine learning techniques



R. Bajaj. Exploiting DSP block capabilities in FPGA high level design flows. PhD Dissertation, NTU, 2017.

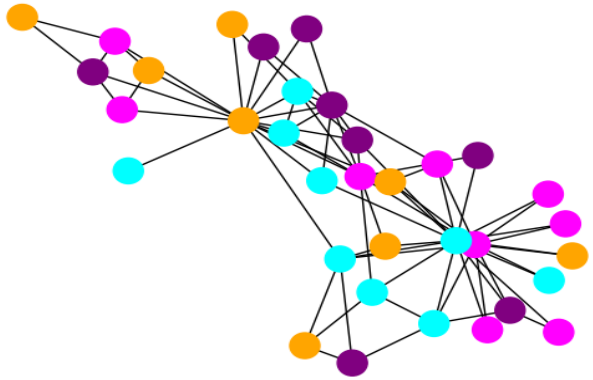
Adder Clusters

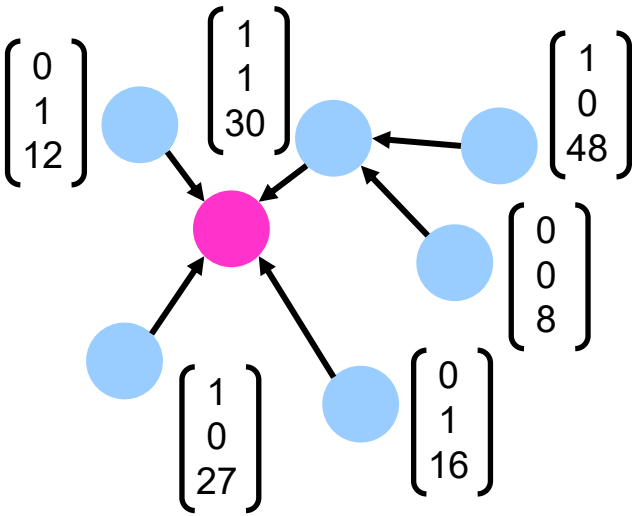
- HLS tools fail to identify many adder clusters
- We propose to learn adder clusters automatically



Learning Operation Mapping Using Graph Learning

- Models for learning representations from complex data, e.g., CNN, RNN, etc.
  - Apply to regular patterns or sequences of data
- Graphs are highly irregular
  - Inconvenient for feature engineering
- Graph Learning
  - Apply ML models to graph-structured data
  - Node embedding: learn low-dimensional representations
  - Neighborhood aggregation methods [1-3]

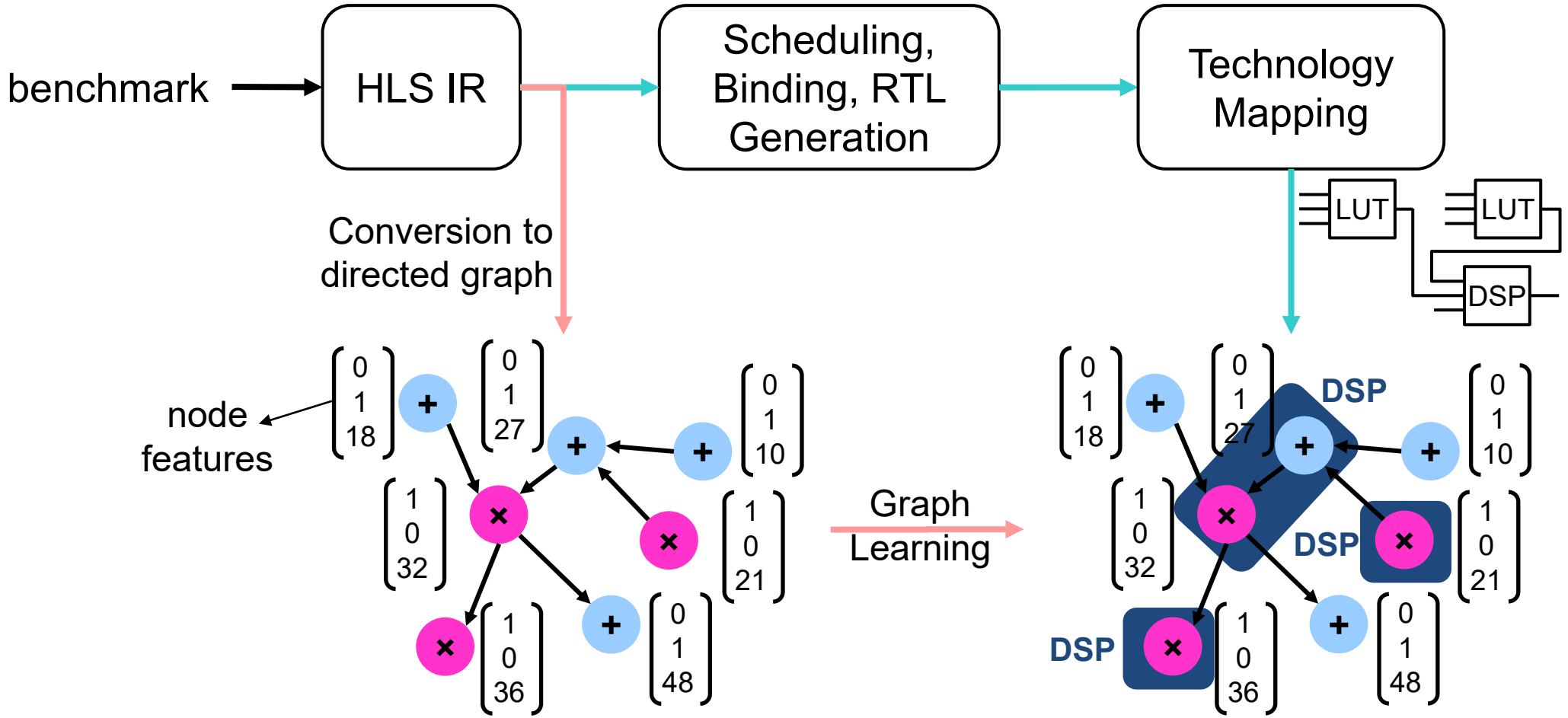




[1] Franco Scarselli, et al. The graph neural network model. IEEE Transactions on Neural Networks, 2009.
[2] Will Hamilton, et al. Inductive representation learning on large graphs. NeurIPS, 2017.
[3] Petar Velickovic, et al. Graph attention networks. arXiv:1710.10903, 2017.

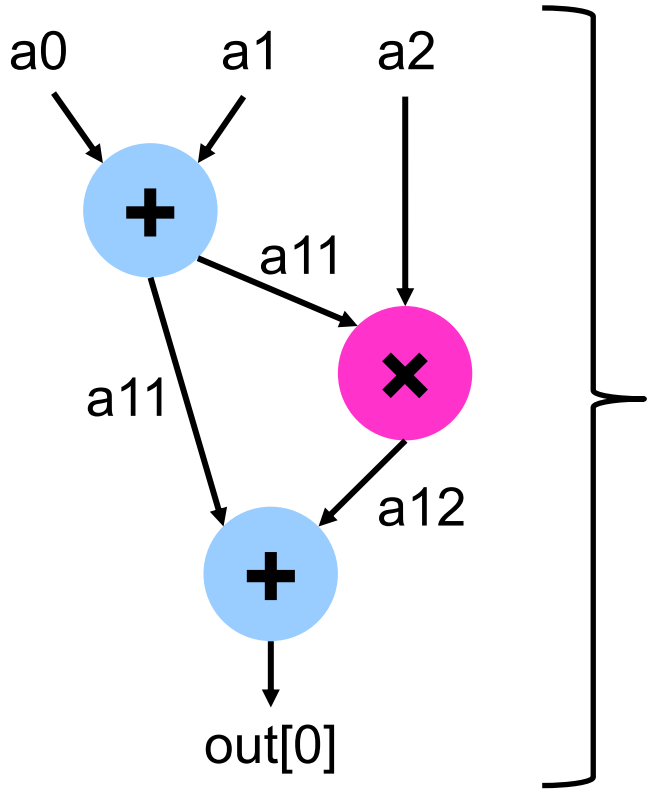
Learning Operation Mapping – Our Approach

- Formulate graph learning on dataflow graphs to learn mapping patterns
- Automatically extract correlation between HLS operations and netlist objects



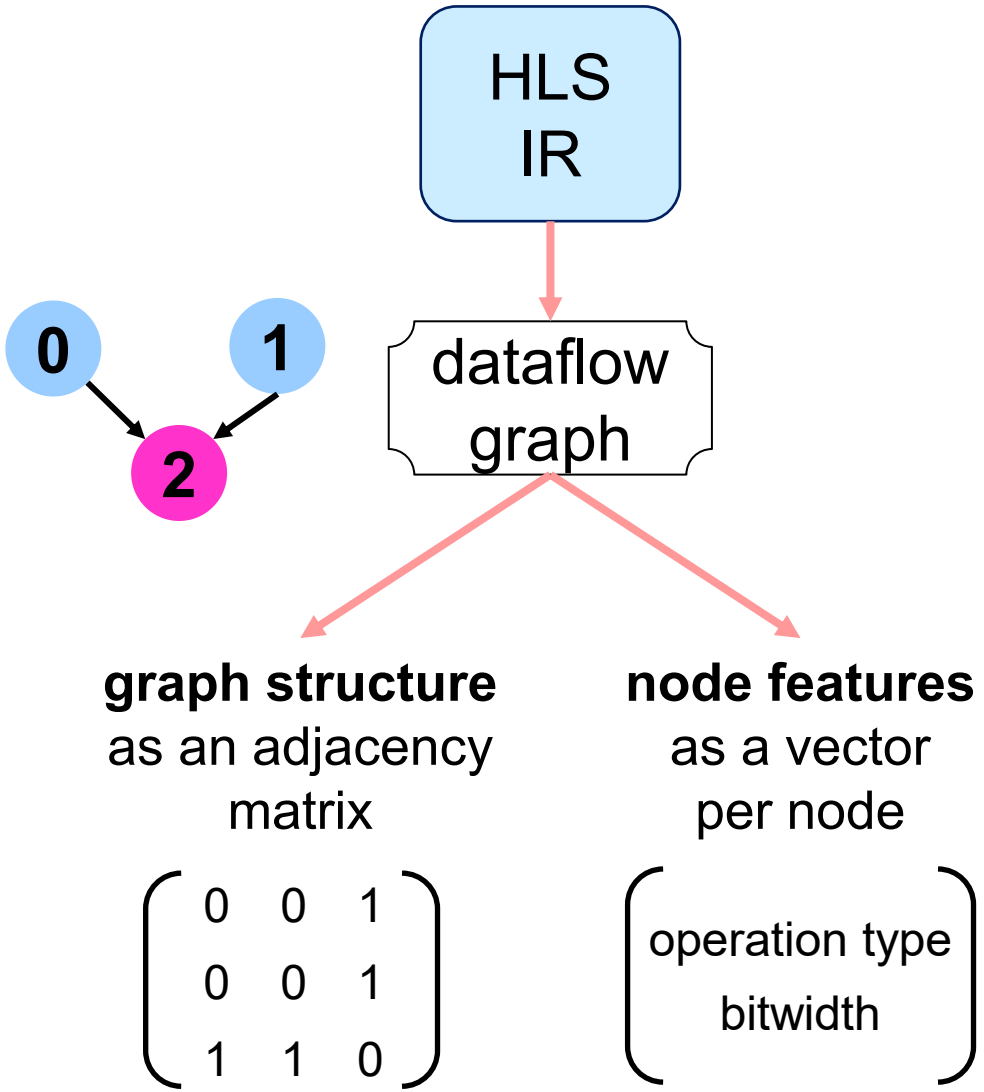
Data Collection – Microbenchmark Generation

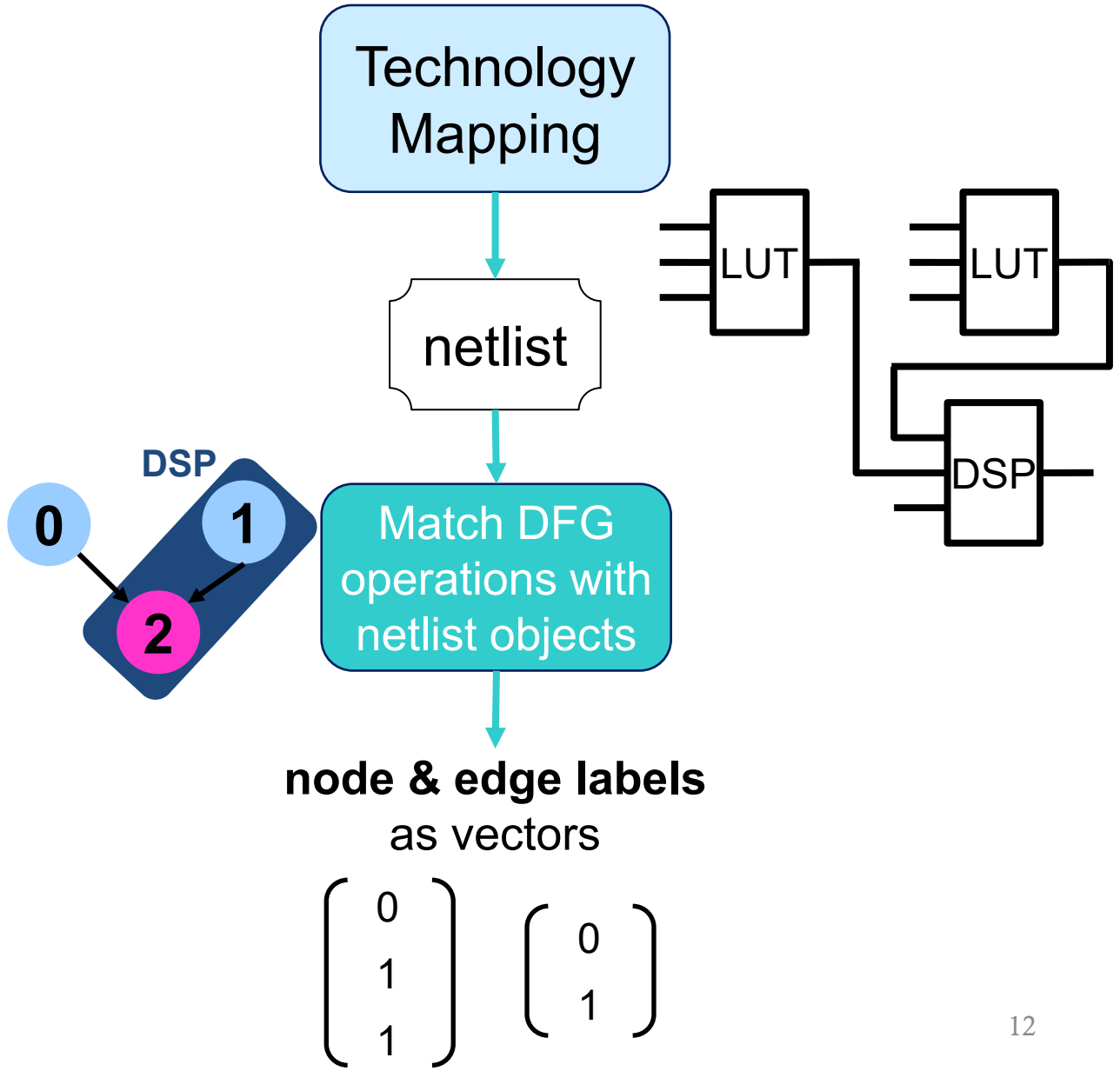
- Each microbenchmark
  - 20 operations in total, 4–8 multiplication operations
  - 8–12 input arguments




```
void DUT(int a0, int a1, int a2,
  int& out){
        int a11;
        int a12;
        a11 = a0 + a1;
        a12 = a11 * a2;
        out[0] = a11 + a12;
}
```

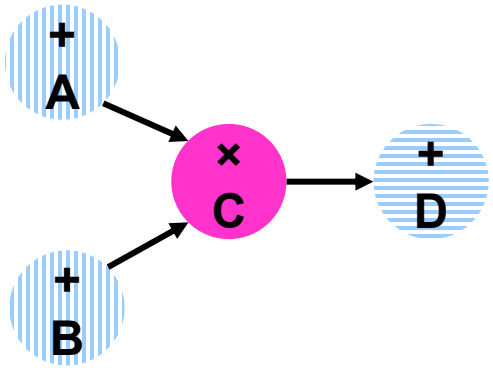
Data Collection – Features and Labels



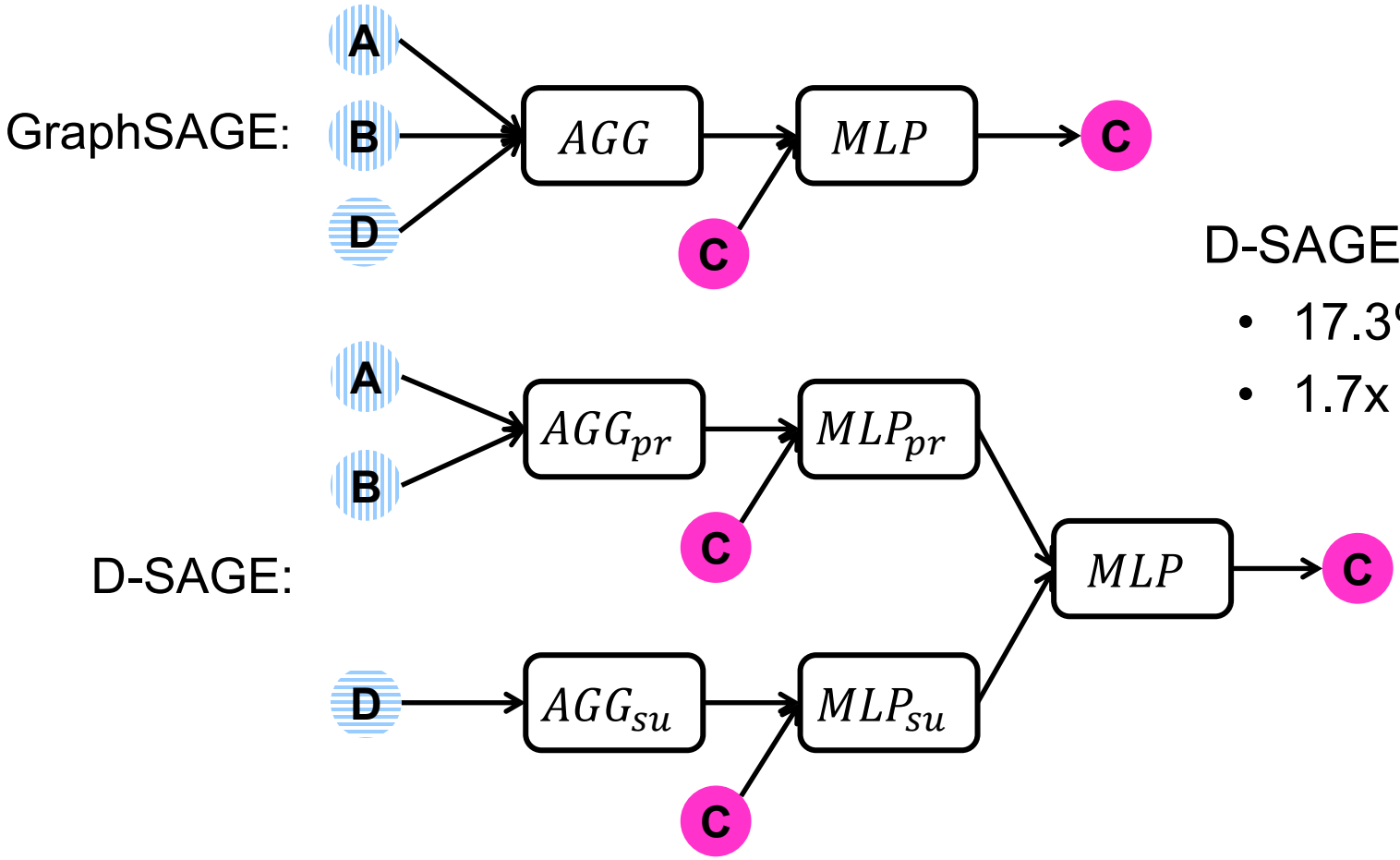


Our GNN Model: D-SAGE

- Extended GraphSAGE [1] to support directed graphs
- Our model D-SAGE can distinguish between pre-adder and post-adder



dataflow graph



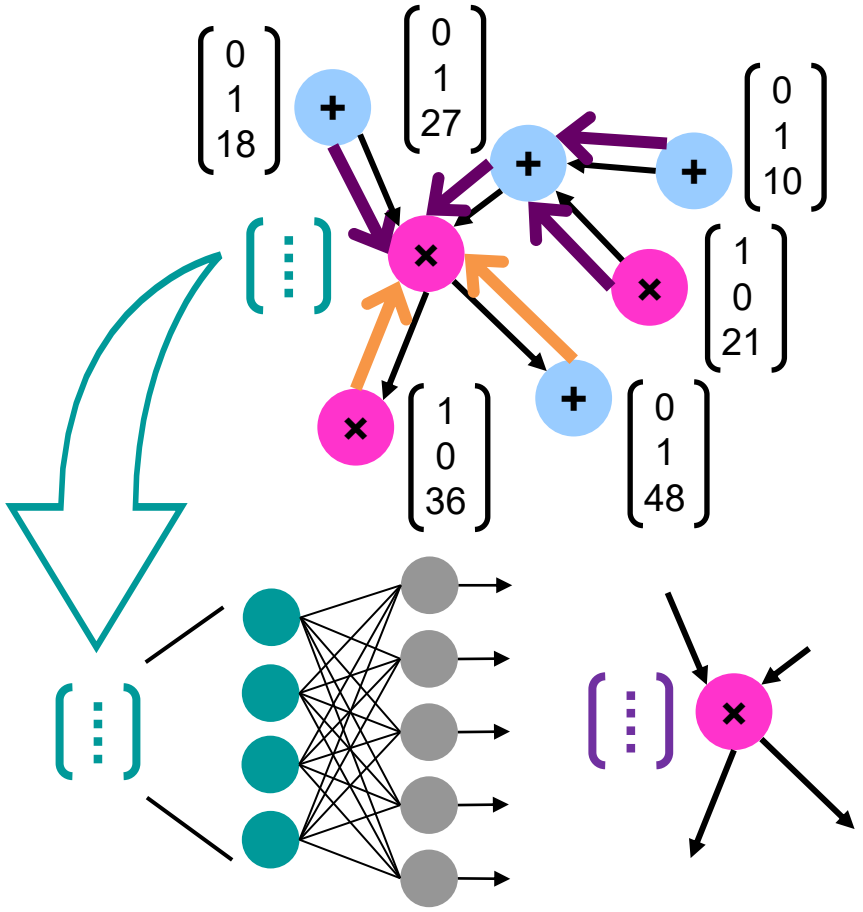
D-SAGE achieves

- 17.3% better accuracy
- 1.7x faster convergence

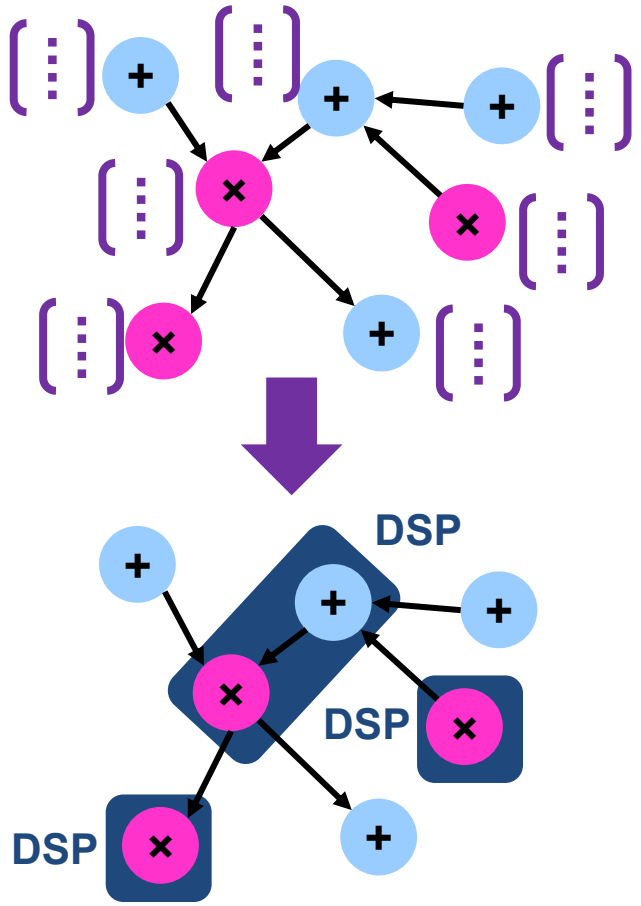
[1] Will Hamilton, et al. Inductive representation learning on large graphs. NeurIPS, 2017.

Our GNN Model: D-SAGE

Step 1: Generate node embedding functions



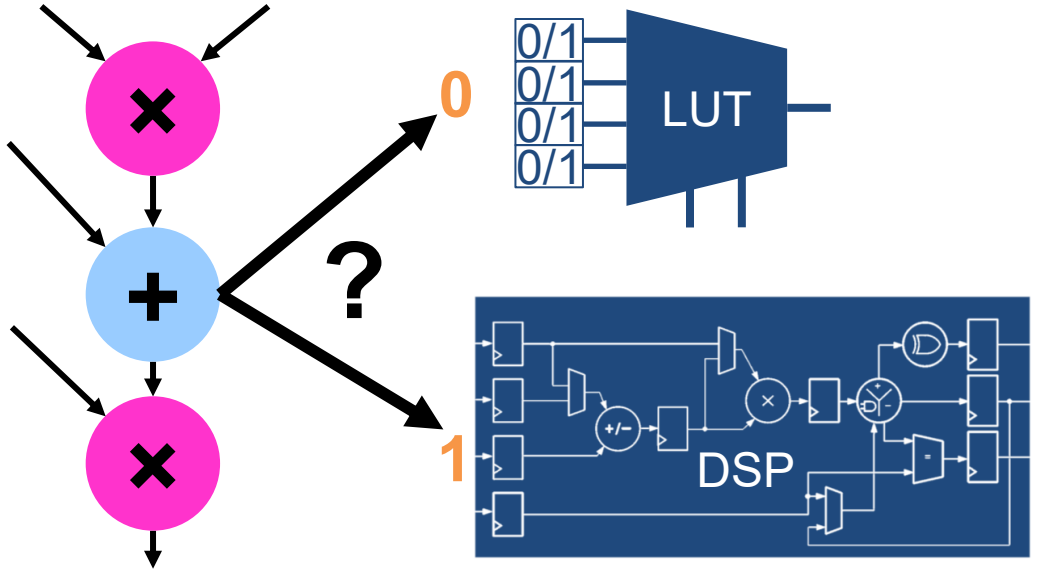
Step 2: Learn node embeddings through supervised learning

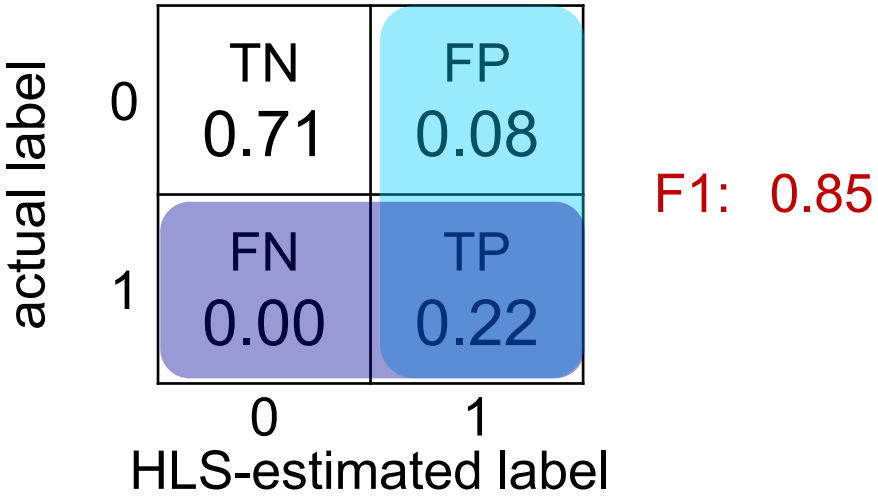


Learning Operation Mapping – DSP Blocks

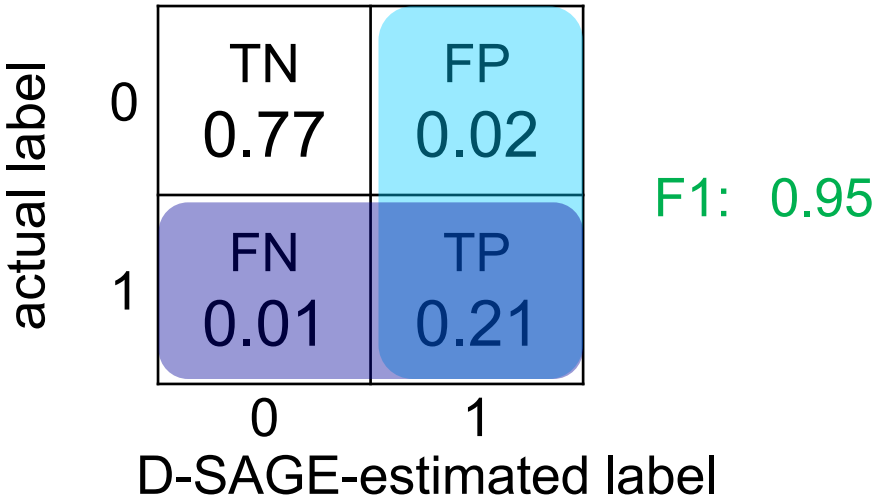
- Virtex UltraScale+ xcvu11p
- Binary node classification
  - Classification of arithmetic operations with respect to DSP mapping

$$node\ label = \begin{cases} 1, if\ maps\ to\ DSP \\ 0, otherwise \end{cases}$$





F1: 0.85

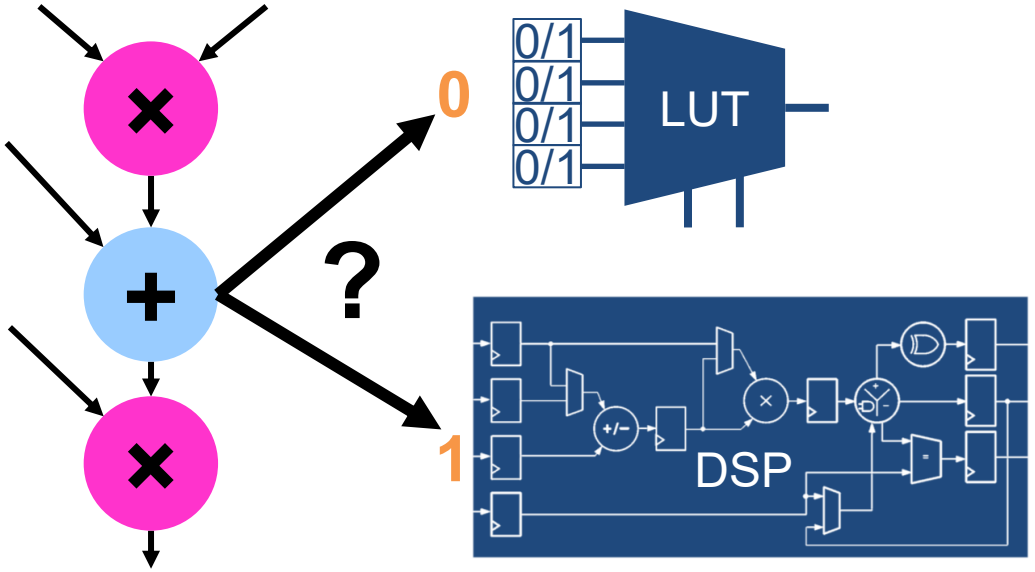


F1: 0.95

Learning Operation Mapping – DSP Blocks

- Train on combinational microbenchmarks, test on realistic designs targeting 250 MHz
  - fir
  - fft
  - gemm
  - md
  - spmv
  - stencil

  } MachSuite



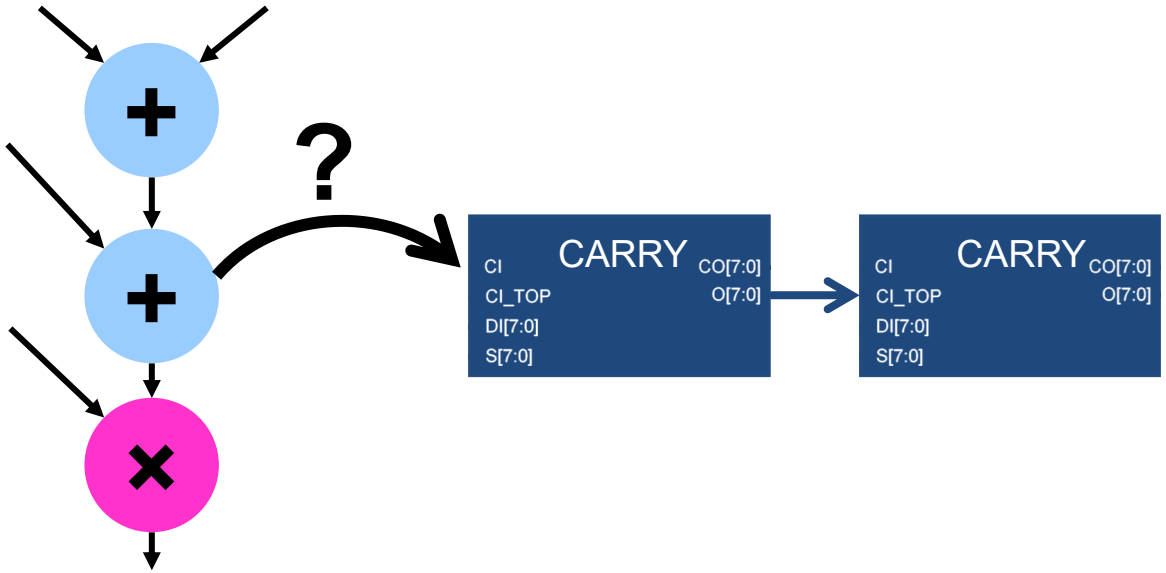
	F1 Score
HLS	0.43
D-SAGE	0.63

Improvement: 47%

Learning Operation Mapping – Carry Chains

- Binary node classification
  - Classification of operations with respect to carry chains

$$node\ label = \begin{cases} 1, if\ maps\ to\ a\ carry\ chain \\ 0, otherwise \end{cases}$$

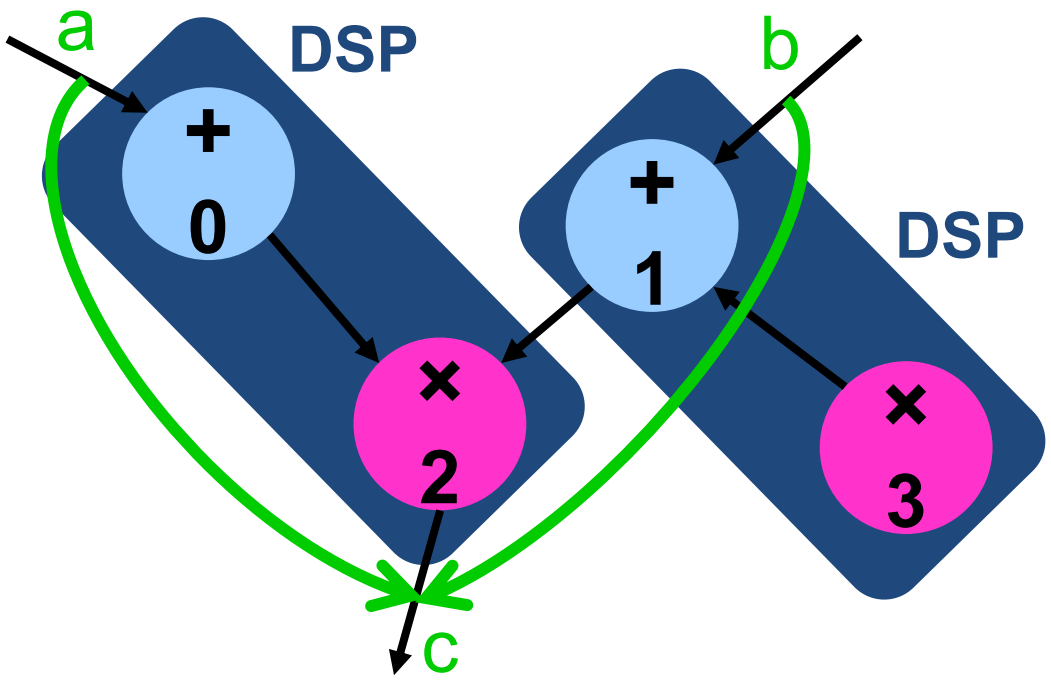


	F1 Score
HLS	0.23
D-SAGE	0.72

Improvement: 213%

Delay Characterization

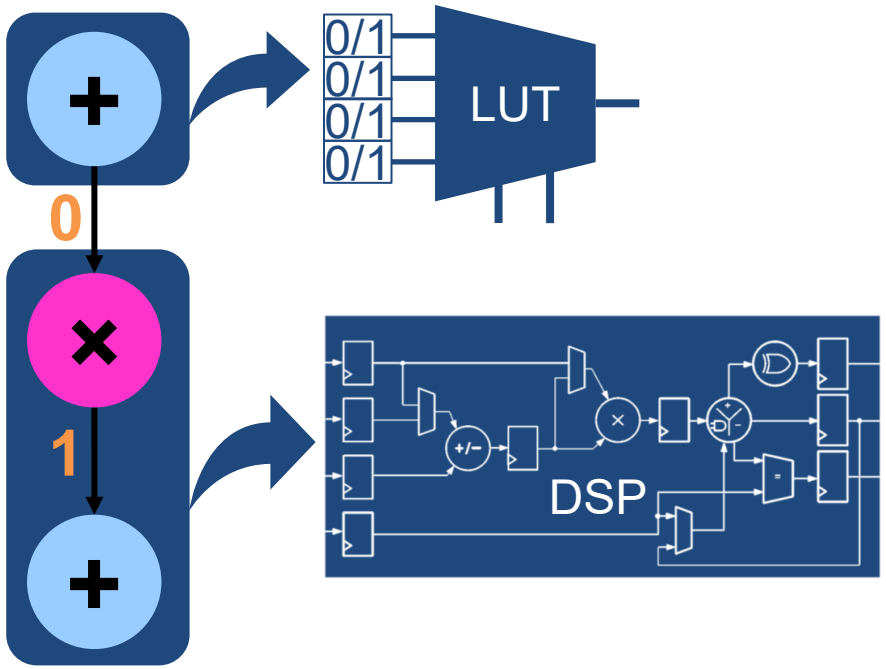
- Are node labels sufficient to infer delay?



Learning Operation Clustering – DSP Blocks

- Binary edge classification
  - Classification of “abstract edges” between operations with respect to DSP clustering

$$edge\ label = \begin{cases} 1, if\ its\ nodes\ are\ clustered \\ 0, otherwise \end{cases}$$

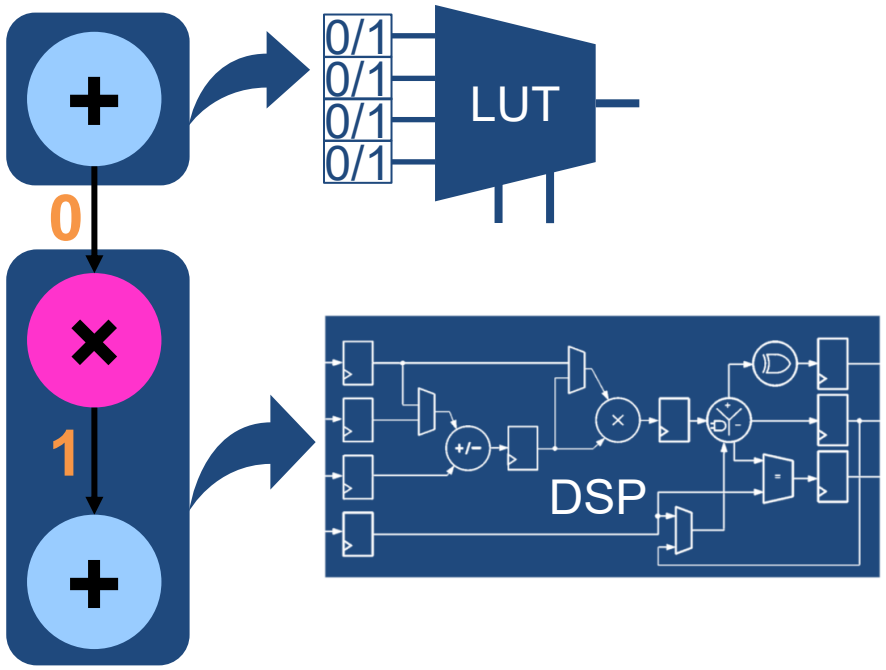


	F1 Score
HLS	0.69
D-SAGE	0.88

Improvement: 28%

Learning Operation Clustering – DSP Blocks

- Train on combinational microbenchmarks, test on realistic designs
  - fir, fft, gemm, md, spmv, stencil
  - Targeting 250 MHz



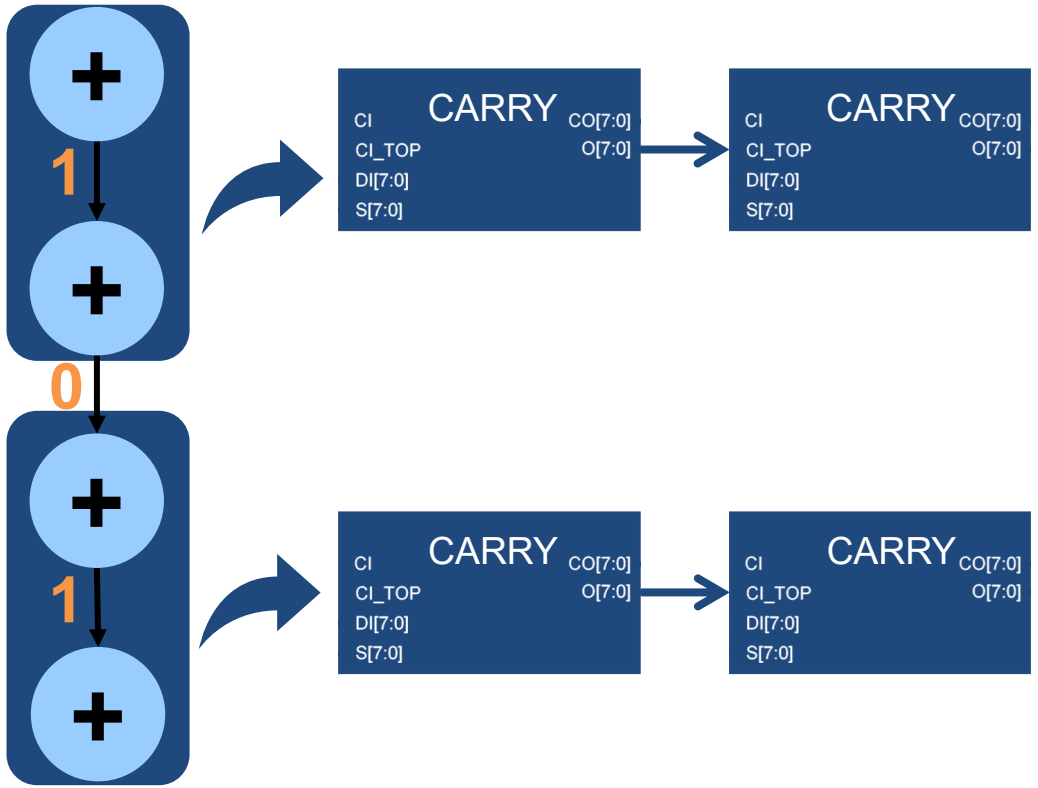
	F1 Score
HLS	0.35
D-SAGE	0.59

Improvement: 69%

Learning Operation Clustering – Carry Chains

- Binary edge classification
  - Classification of “abstract edges” between operations with respect to carry chains

$$edge\ label = \begin{cases} 1, if\ its\ nodes\ are\ clustered \\ 0, otherwise \end{cases}$$

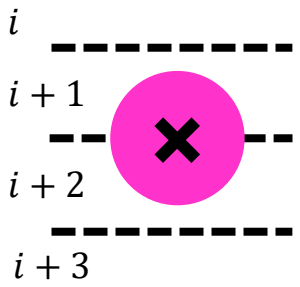


	F1 Score
HLS	0.17
D-SAGE	0.82

Improvement: 382%

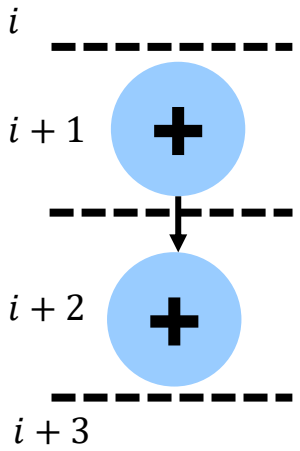
Extension to Pipelined Operations

- Incorporate scheduling information



DSP Mapping	µbenchmarks F1 Score	MachSuite F1 Score
HLS	0.72	0.55
D-SAGE	0.89	0.65

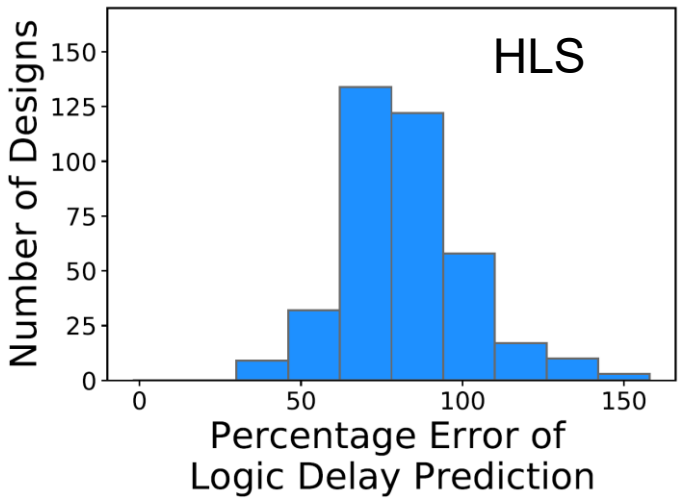
DSP Clustering	µbenchmarks F1 Score	MachSuite F1 Score
HLS	0.71	0.45
D-SAGE	0.84	0.51

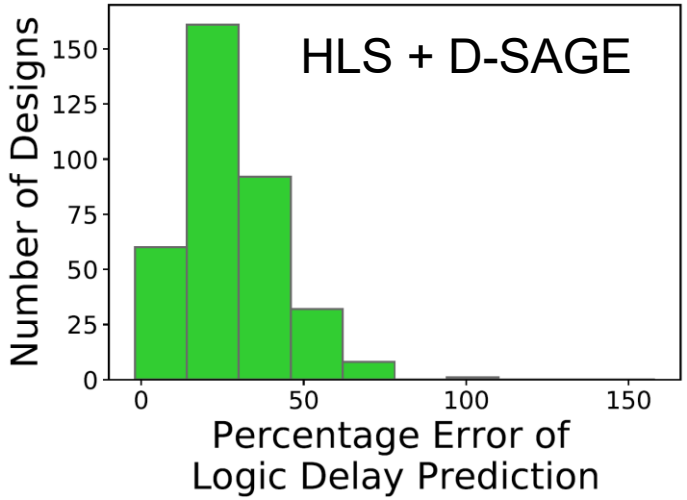


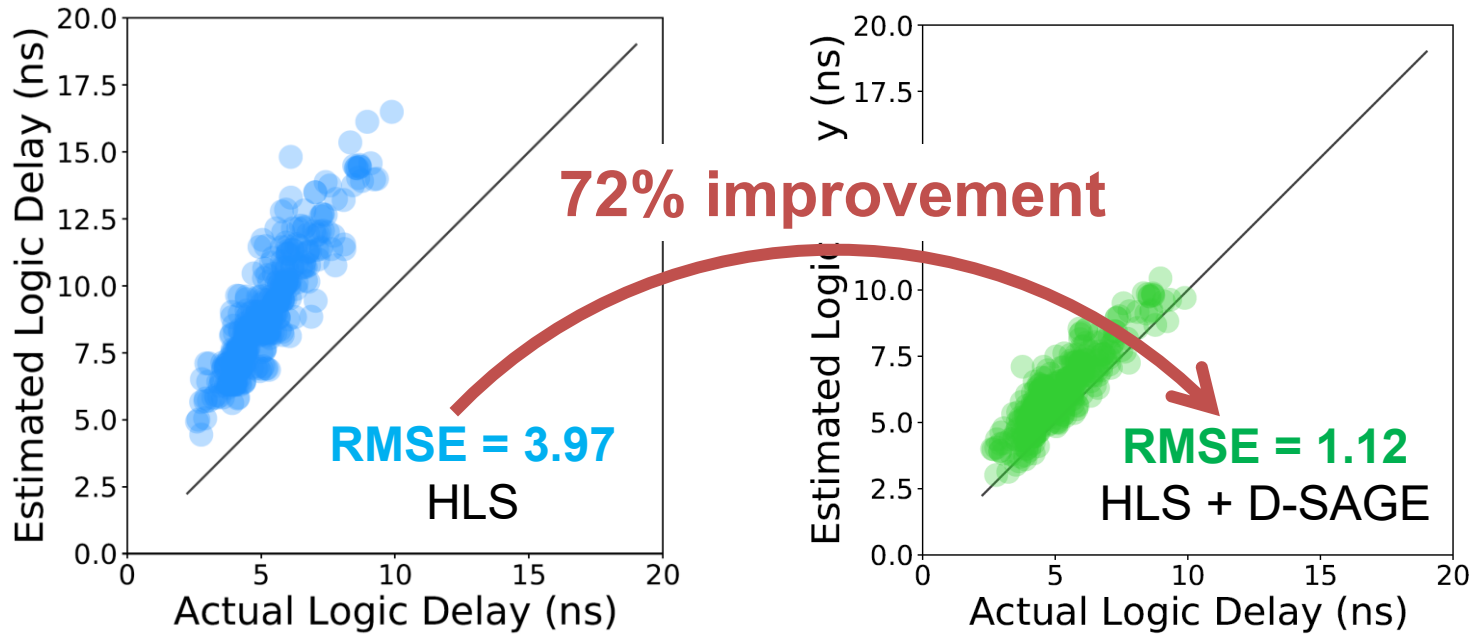
Carry Chain Mapping	µbenchmarks F1 Score
HLS	0.43
D-SAGE	0.73

Carry Chain Clustering	µbenchmarks F1 Score
HLS	0.28
D-SAGE	0.59

Accurate Delay Prediction for HLS







μbenchmarks	RRSE	
	Logic Delay	Datapath Delay
HLS	3.32	2.56
QuickEst	0.34	0.41
HLS + D-SAGE	0.83	0.35
QuickEst + D-SAGE	0.28	0.28

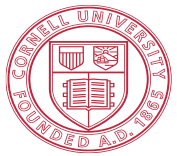
Realistic designs	RRSE	
	Logic Delay	Datapath Delay
HLS	2.19	2.22
QuickEst	1.92	2.44
HLS + D-SAGE	0.82	0.83
QuickEst + D-SAGE	0.89	1.20

Accurate Operation Delay Prediction for FPGA HLS Using Graph Neural Networks

Ecenur Ustun*, Chenhui Deng*, Debjit Pal, Zhijing Li, Zhiru Zhang

Electrical and Computer Engineering, Cornell University

Thank you!



Cornell University

CSL