StreamTensor: A Compiler from PyTorch to FPGA for AI/ML Applications

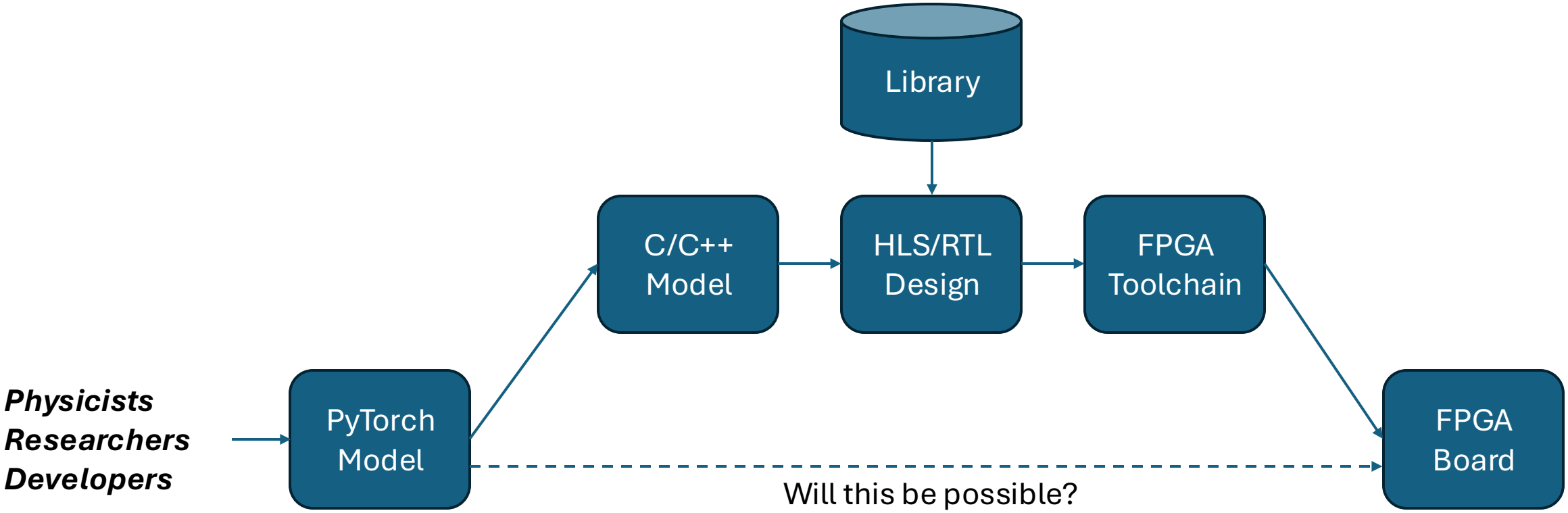
Hanchen Ye[12] (hanchen8@illinois.edu)

Deming Chen[12] (dchen@illinois.edu)

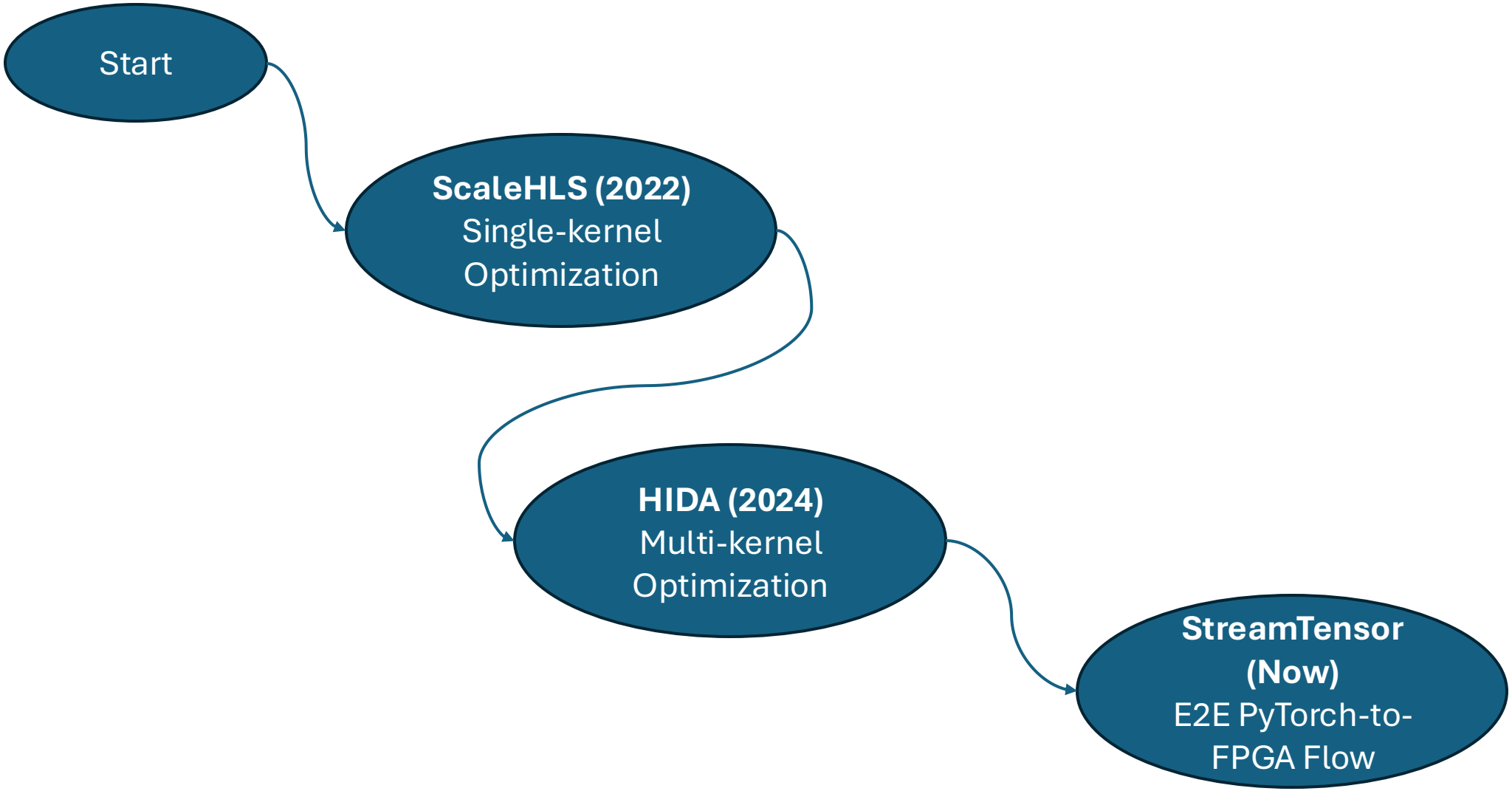
[1]*University of Illinois Urbana-Champaign* [2]*Inspirit IoT*

Hardware and Algorithm Co-development (HAC) Research Highlight
2024 NSF HDR Ecosystem Conference

Application Development on FPGA



Path to an E2E PyTorch-to-FPGA Flow



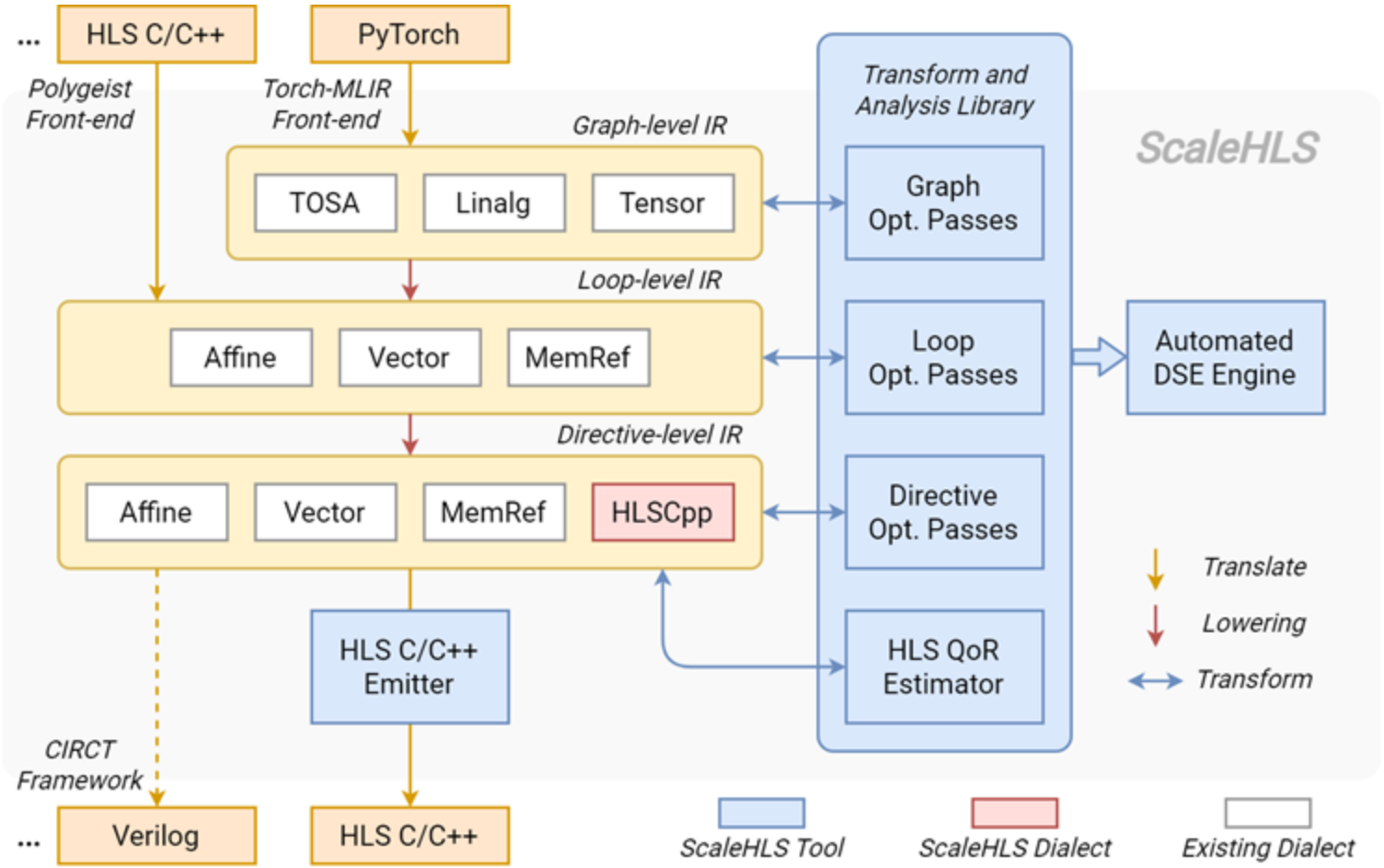
ScaleHLS: Single-kernel Optimization

[1] Scalehls: A new scalable high-level synthesis framework on multi-level intermediate representation, HPCA'22

[2] ScaleHLS: a scalable high-level synthesis framework with multi-level transformations and optimizations, DAC'22

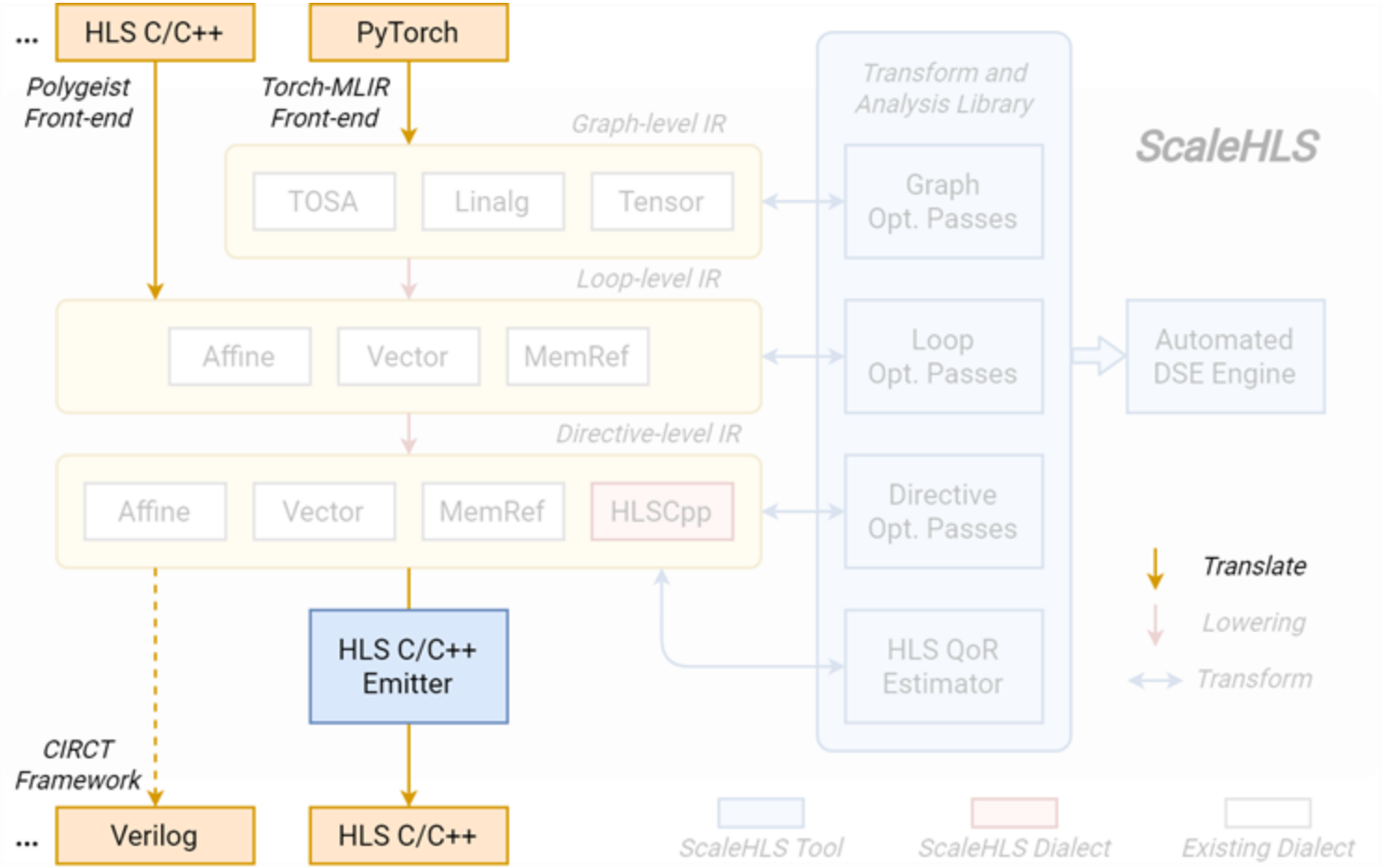
[3] High-level synthesis for domain specific computing, ISPD'23

Framework Overview



[1] Polygeist: https://github.com/wsmoses/Polygeist [2] Torch-MLIR: https://github.com/llvm/torch-mlir [3] CIRCT: https://github.com/llvm/circt

Front-end and Back-end



Inputs

C/C++ Polygeist [1]

PyTorch Torch-MLIR [2]

Outputs

C/C++ C/C++ Emitter

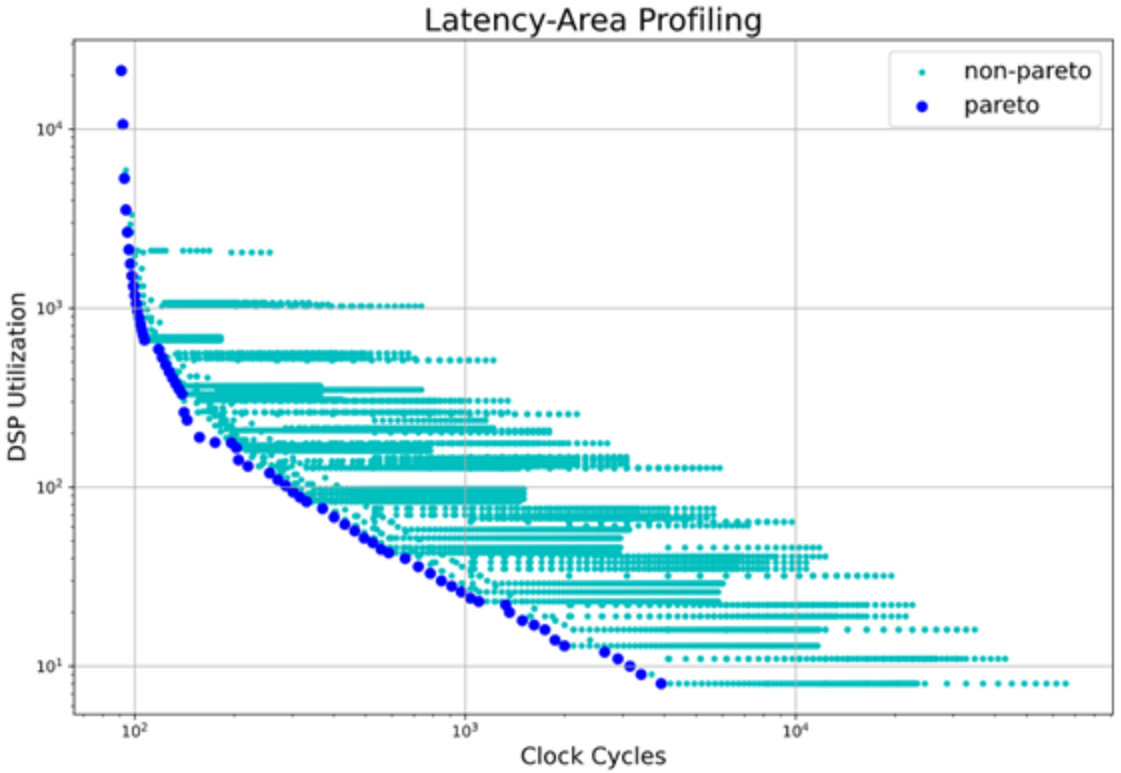
Verilog CIRCT [3] (work-in-progress)

[1] Polygeist: https://github.com/wsmoses/Polygeist

[2] Torch-MLIR: https://github.com/llvm/torch-mlir

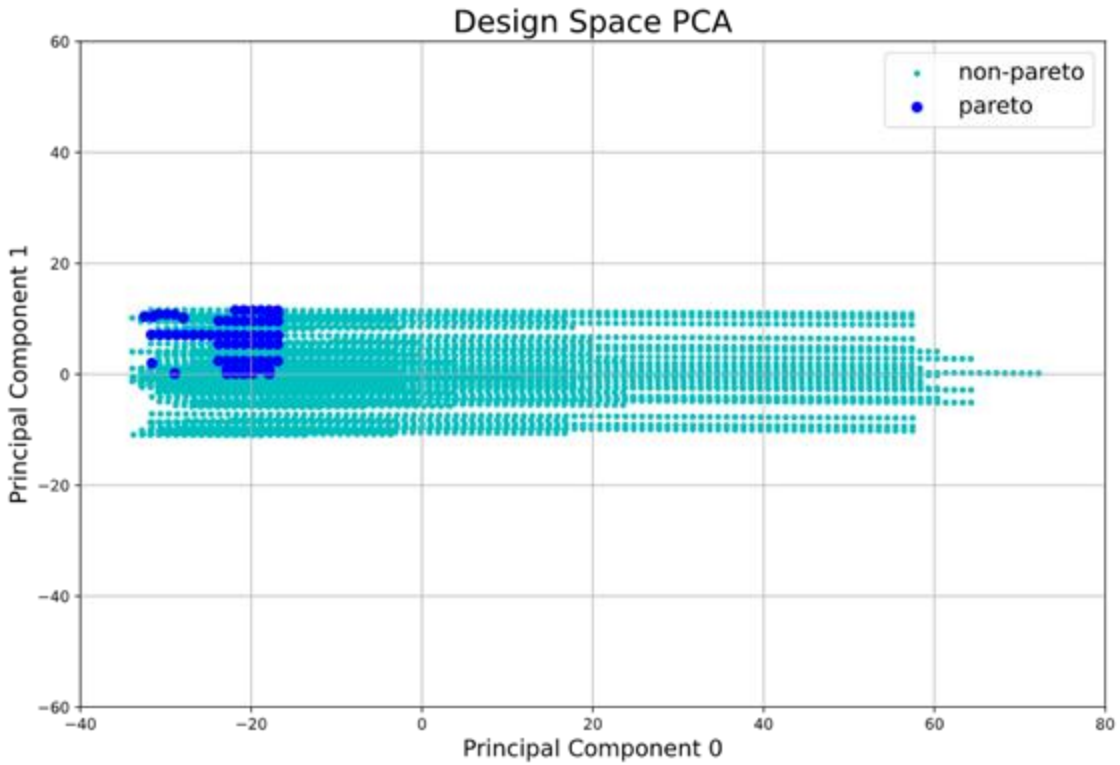
[3] CIRCT: https://github.com/llvm/circt

Single-kernel Design Space Exploration



Pareto frontier of a GEMM kernel

- Latency and area are profiled for each design point
- Dark blue points are Pareto points
- Loop perfectization, loop order permutation, loop tiling, loop pipelining, and array partition passes are involved



- Each parameter of a pass becomes one dimension, the original 4-dimensional design space is reduced to two dimensions through PCA
- Pareto points are located at a corner of the design space, the variance of Pareto points is much smaller than the overall variance

Polybench Results

Kernel	Prob. Size	Speedup	LP	RVB	Perm. Map	Tiling Sizes	Pipeline II	Array Partition
BICG	4096	41.7×	No	No	[1, 0]	[16, 8]	43	A:[8, 16], s:[16], q:[8], p:[16], r:[8]
GEMM	4096	768.1×	Yes	No	[1, 2, 0]	[8, 1, 16]	3	C:[1, 16], A:[1, 8], B:[8, 16]
GESUMMV	4096	199.1×	Yes	No	[1, 0]	[8, 16]	9	A:[16, 8], B:[16, 8], tmp:[16], x:[8], y:[16]
SYR2K	4096	384.0×	Yes	Yes	[1, 2, 0]	[8, 4, 4]	8	C:[4, 4], A:[4, 8], B:[4, 8]
SYRK	4096	384.1×	Yes	Yes	[1, 2, 0]	[64, 1, 1]	3	C:[1, 1], A:[1, 64]
TRMM	4096	590.9×	Yes	Yes	[1, 2, 0]	[4, 4, 32]	13	A:[4, 4], B:[4, 32]

DSE results of PolyBench-C computation kernels

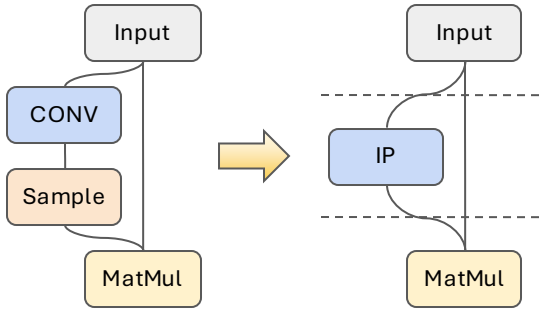
1. The target platform is Xilinx XC7Z020 FPGA, which is an edge FPGA with 4.9 Mb memories, 220 DSPs, and 53,200 LUTs. The data types of all kernels are single-precision floating-points.
2. Among all six benchmarks, a **speedup** ranging from 41.7× to 768.1× is obtained compared to the baseline design, which is the original computation kernel from PolyBench-C without the optimization of DSE.
3. **LP** and **RVB** denote Loop Perfectization and Remove Variable Bound, respectively.
4. In the Loop Order Optimization (**Perm. Map**), the i-th loop in the loop nest is permuted to location $PermMap\,[i]$, where locations are from the outermost loop to inner.

HIDA: Multi-kernel Optimization

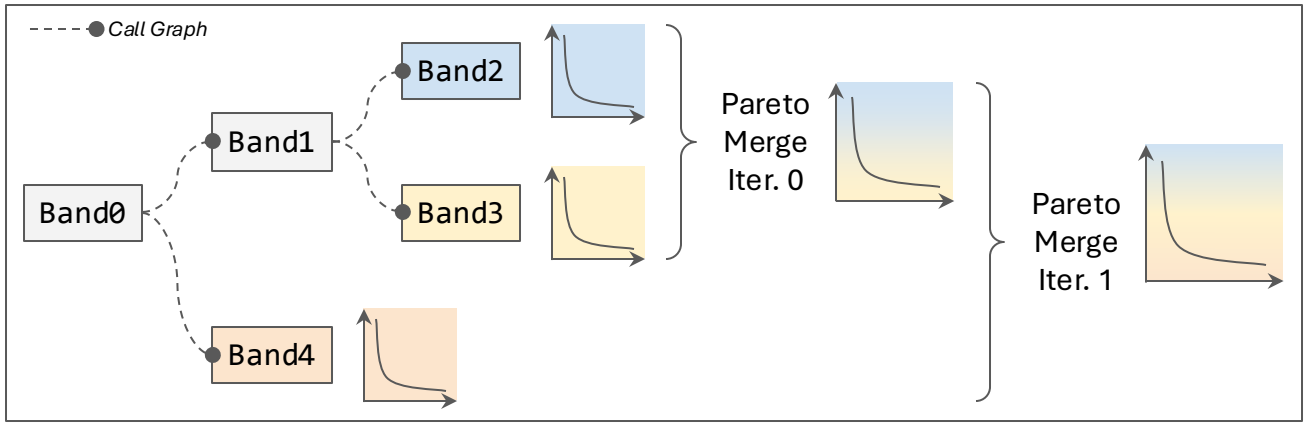
[1] HIDA: A Hierarchical Dataflow Compiler for High-Level Synthesis, ASPLOS’24

[2] ScaleFlow: High-Level Synthesis for Large Dataflow Applications, TECHCON’23

Multi-kernel Optimization in ScaleHLS



Graph Optimizations



Step (2) Global multi-kernel Pareto curving merging

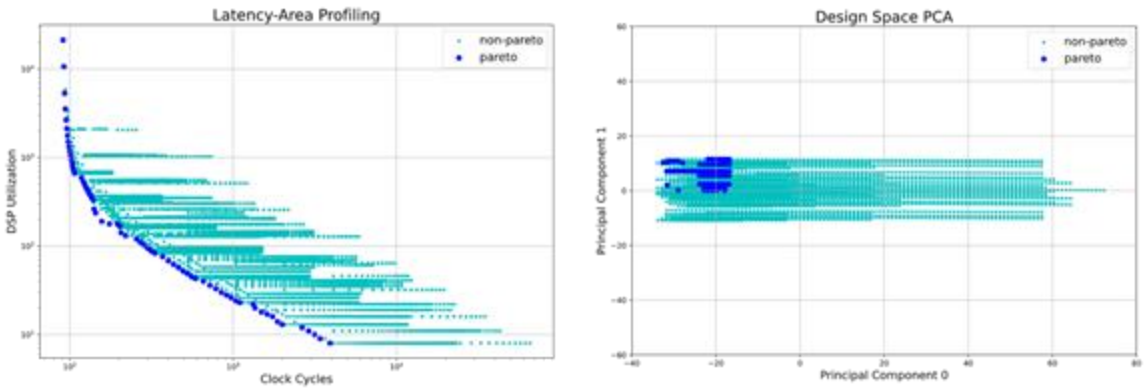
```
for (int i = 0; i < 32; i++) {
  for (int j = 0; j < 32; j++) {
    C[i][j] *= beta;
    for (int k = 0; k < 32; k++) {
      C[i][j] += alpha * A[i][k] * B[k][j];
} } }
```

Loop Optimizations

```
for (int k = 0; k < 32; k++) {
  for (int i = 0; i < 32; i++) {
    for (int j = 0; j < 32; j++) {
      if (k == 0)
        C[i][j] *= beta;
      C[i][j] += alpha * A[i][k] * B[k][j];
} } }
```

Directive Optimizations

```
for (int k = 0; k < 32; k++) {
  for (int i = 0; i < 32; i++) {
    for (int j = 0; j < 32; j++) {
#pragma HLS pipeline
      if (k == 0)
        C[i][j] *= beta;
      C[i][j] += alpha * A[i][k] * B[k][j];
} } }
```

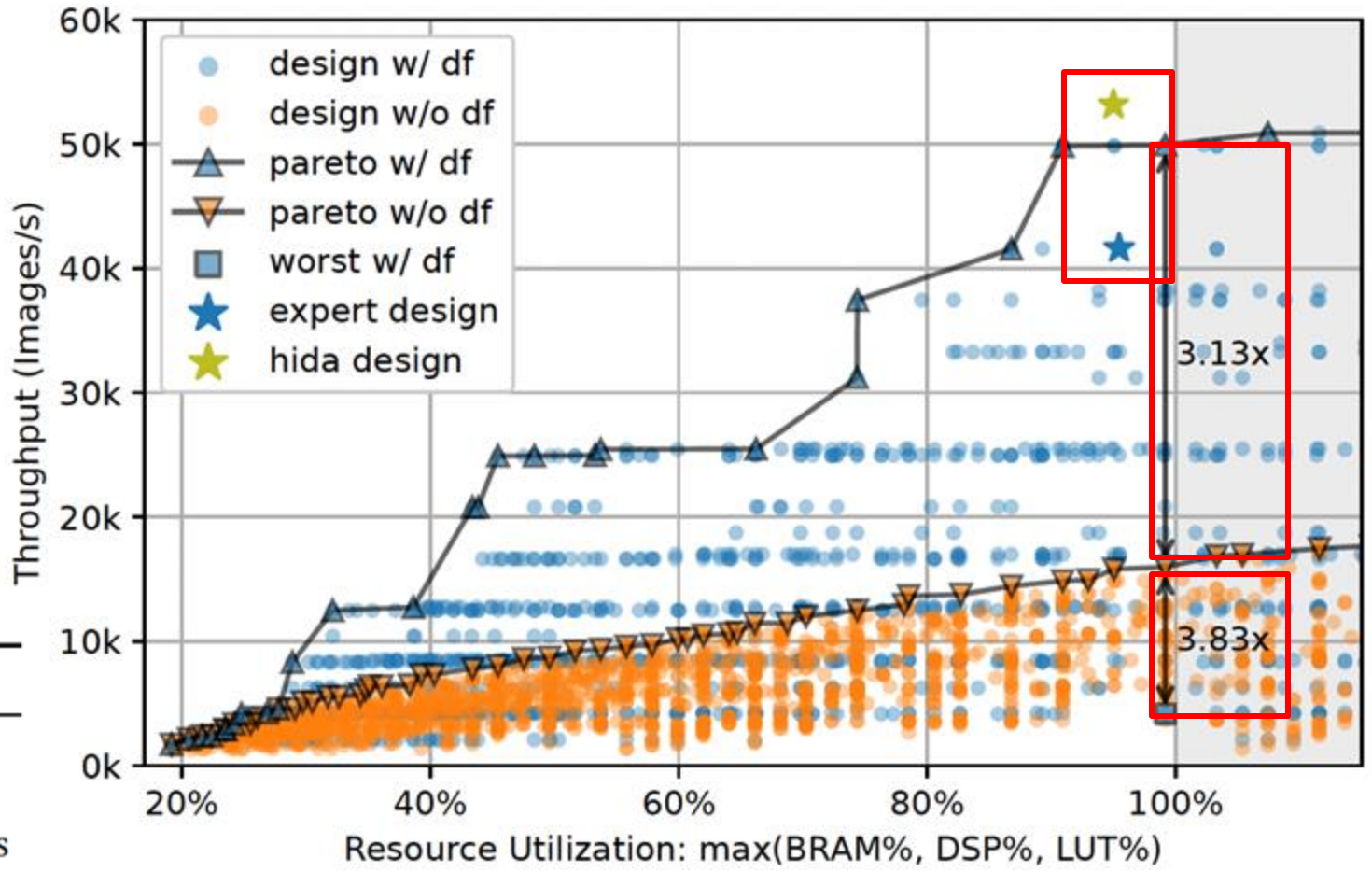


Step (1) Local single-kernel *loop* and *directive* DSE

Multi-kernel Optimization is Important but Difficult

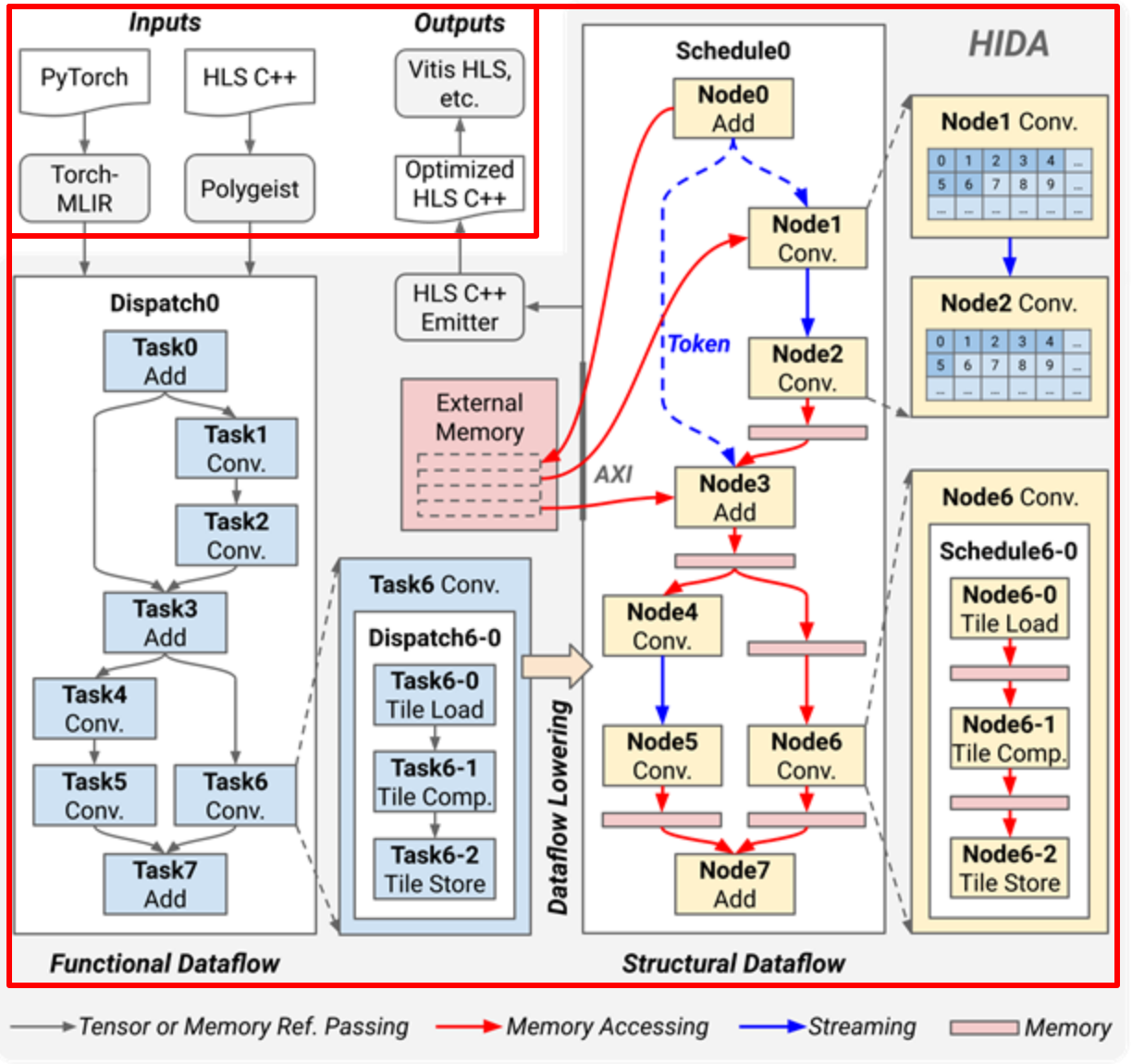
- Dataflow designs are Pareto-dominating
- Dataflow cannot guarantee a good trade-off
- Dataflow design space is difficult to comprehend
- Automated tool outperforms exhaustive search

	Expert	Exhaustive	HIDA
Resource Util.	95.5%	99.2%	95.0%
Throu. (Imgs/s)	41.6k	49.9k	53.2k
Develop Cycle	40 hours	210 hours	9.9 mins



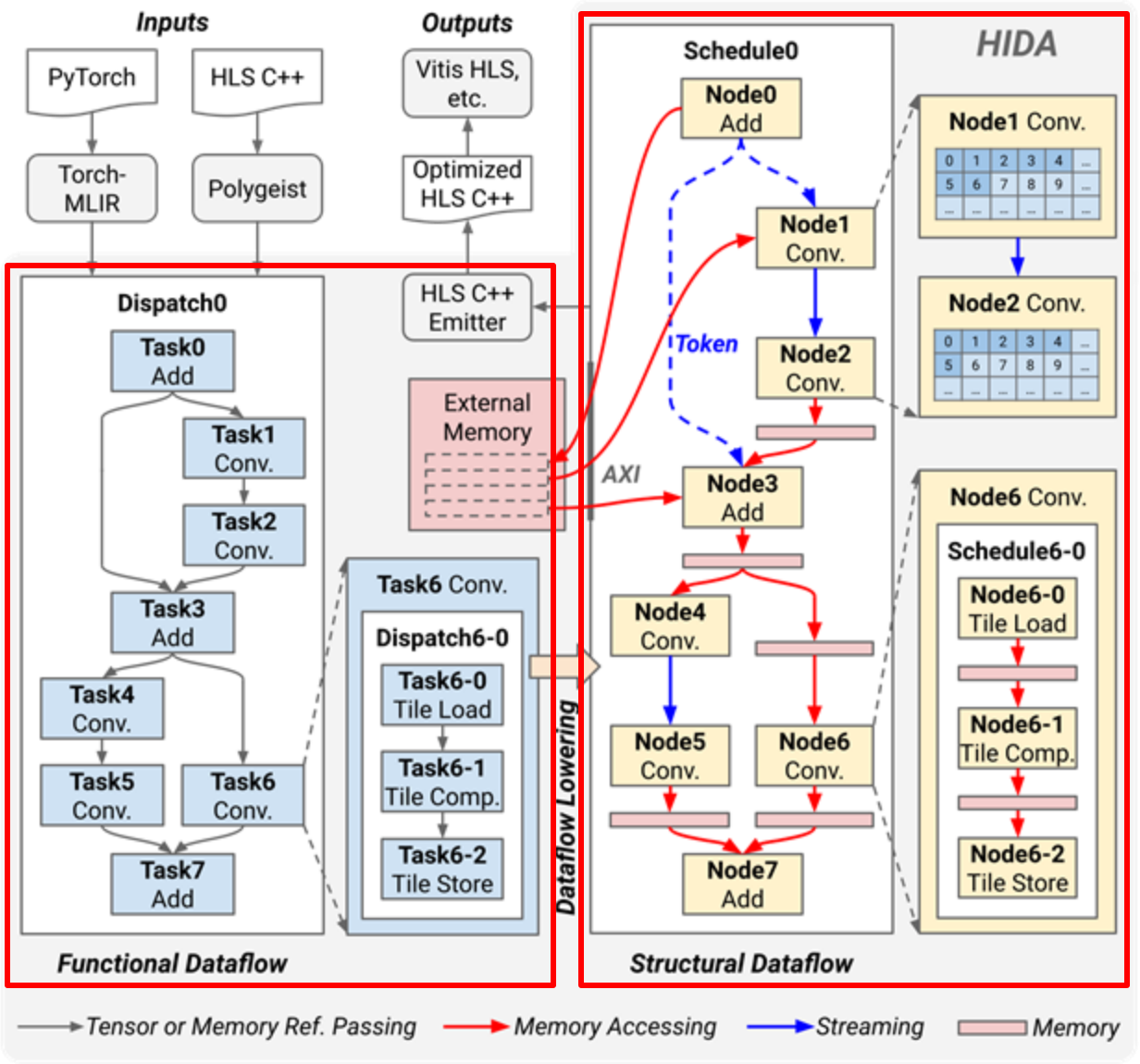
Dataflow = Coarse-grained Pipeline between Kernels

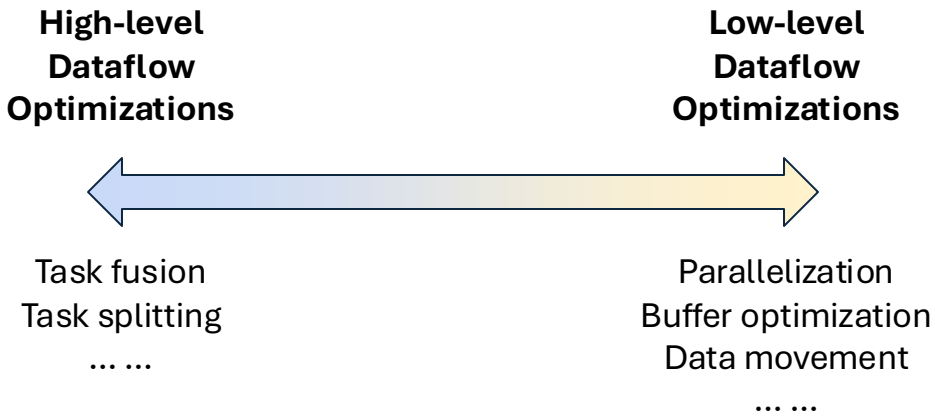
Framework Overview



- **PyTorch** or **C/C++** as input
- Optimized **C++** dataflow design as output
- **MLIR-based** dataflow intermediate representation (IR), optimization, and code-generation

Two-level Dataflow Representation





Two-level dataflow representation

- Functional dataflow
  - Capture high-level dataflow characteristics
  - Efficient dataflow manipulation
- Structural dataflow
  - Capture low-level micro-architectures
  - Efficient scheduling and parallelization

Neural Networks Results

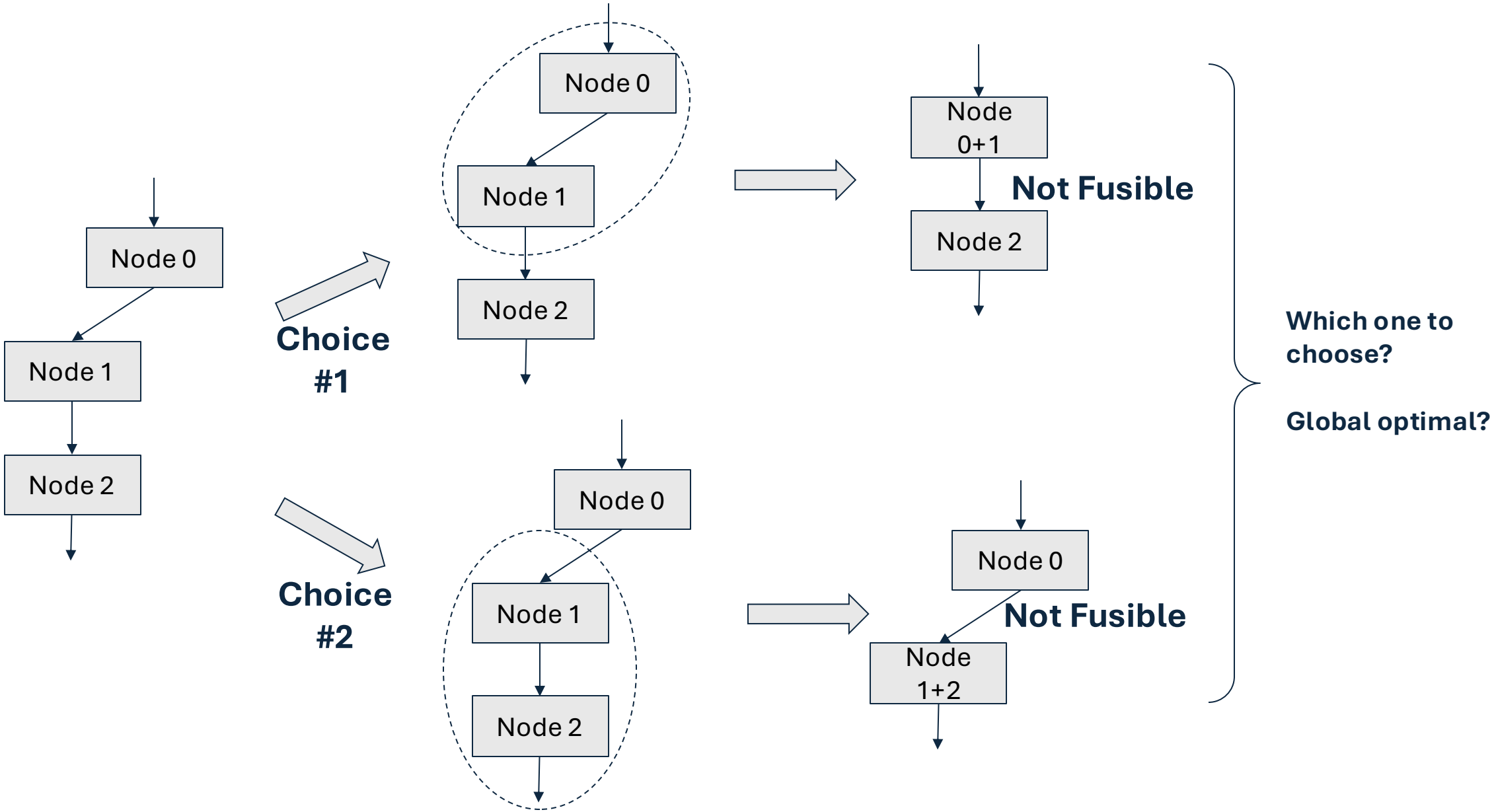
Model	HIDA Compile Time (s)	LUT Number	DSP Number	Throughput (Samples/s)*			DSP Efficiency*		
				HIDA	DNNBuilder [75]	ScaleHLS [68]	HIDA	DNNBuilder [75]	ScaleHLS [68]
ResNet-18	83.1	142.1k	667	45.4	-	3.3 (13.88×)	73.8%	-	5.2% (14.24×)
MobileNet	110.8	132.9k	518	137.4	-	15.4 (8.90×)	75.5%	-	9.6% (7.88×)
ZFNet	116.2	103.8k	639	90.4	112.2 (0.81×)	-	82.8%	79.7% (1.04×)	-
VGG-16	199.9	266.2k	1118	48.3	27.7 (1.74×)	6.9 (6.99×)	102.1%	96.2% (1.06×)	18.6% (5.49×)
YOLO	188.2	202.8k	904	33.7	22.1 (1.52×)	-	94.3%	86.0% (1.10×)	-
MLP	40.9	21.0k	164	938.9	-	152.6 (6.15×)	90.0%	-	17.6% (5.10×)
Geo. Mean	**108.7**				**1.29×**	**8.54×**		**1.07×**	**7.49×**

* Numbers in () show throughput/DSP efficiency improvements of HIDA over others.

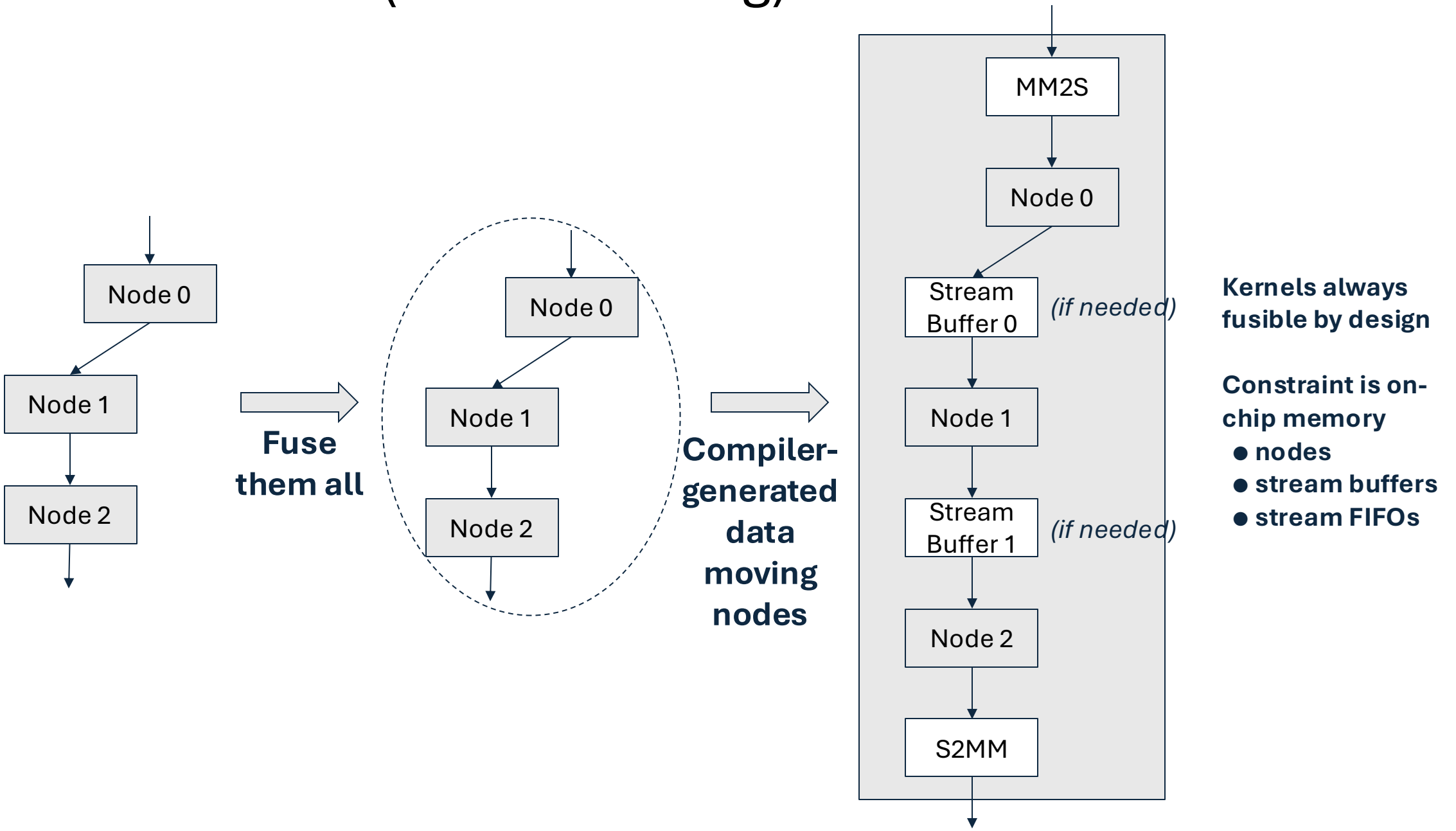
StreamTensor: Streaming-based Multi-kernel Dataflow

Working in progress... Stay tuned!

Kernel Fusion (Traditional)



Kernel Fusion (with Streaming)

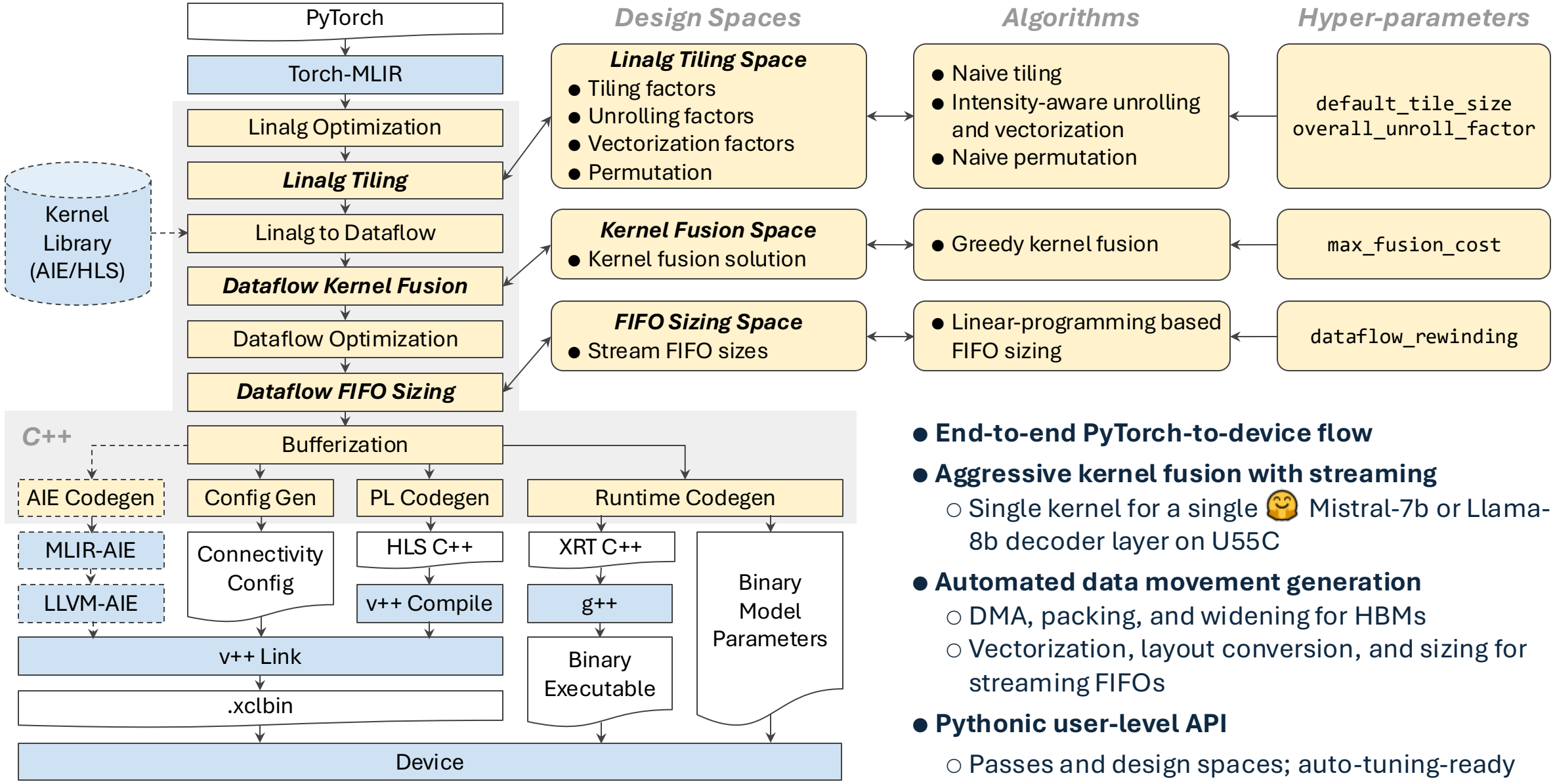


Kernels always fusible by design

Constraint is on-chip memory

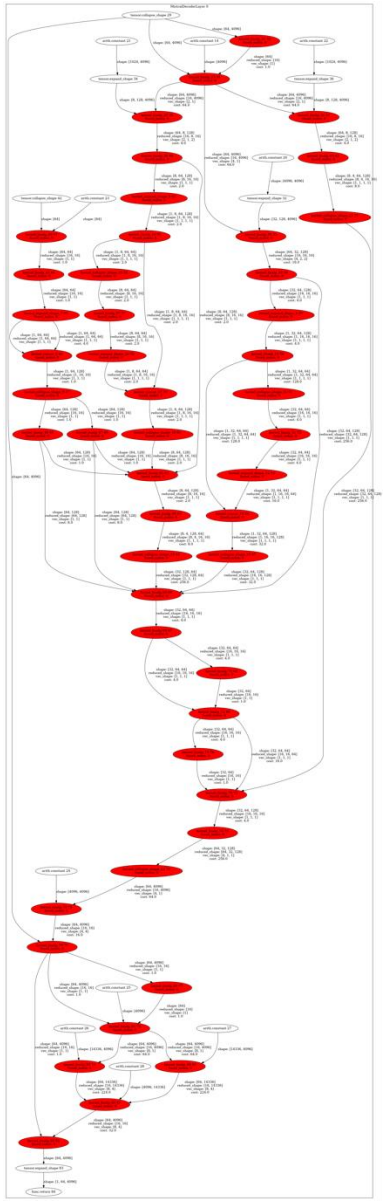
- **nodes**
- **stream buffers**
- **stream FIFOs**

Framework Overview



- **End-to-end PyTorch-to-device flow**
- **Aggressive kernel fusion with streaming**
  - Single kernel for a single 🤗 Mistral-7b or Llama-8b decoder layer on U55C
- **Automated data movement generation**
  - DMA, packing, and widening for HBMs
  - Vectorization, layout conversion, and sizing for streaming FIFOs
- **Pythonic user-level API**
  - Passes and design spaces; auto-tuning-ready

Mistral3-7B: From PyTorch to Board



Single Decoder Layer

Vocabulary size: 32000
Hidden size=4096
Intermediate size=14336
Number of hidden layers=32
Number of attention heads=32

AMD U55C FPGA

- All 48 kernels are fused into one single kernel
- All fused kernels are executed in a coarse-grained dataflow pipeline

Resource	Utilization
BRAM (16kb RAM)	3176
DSP	928
FF	261667
LUT	358433
URAM (256kb RAM)	779

Thanks!

Hanchen Ye[12] (hanchen8@illinois.edu)

Deming Chen[12] (dchen@illinois.edu)

[1]*University of Illinois Urbana-Champaign* [2]*Inspirit IoT*

Hardware and Algorithm Co-development (HAC) Research Highlight

2024 NSF HDR Ecosystem Conference