SRC @ ICCAD

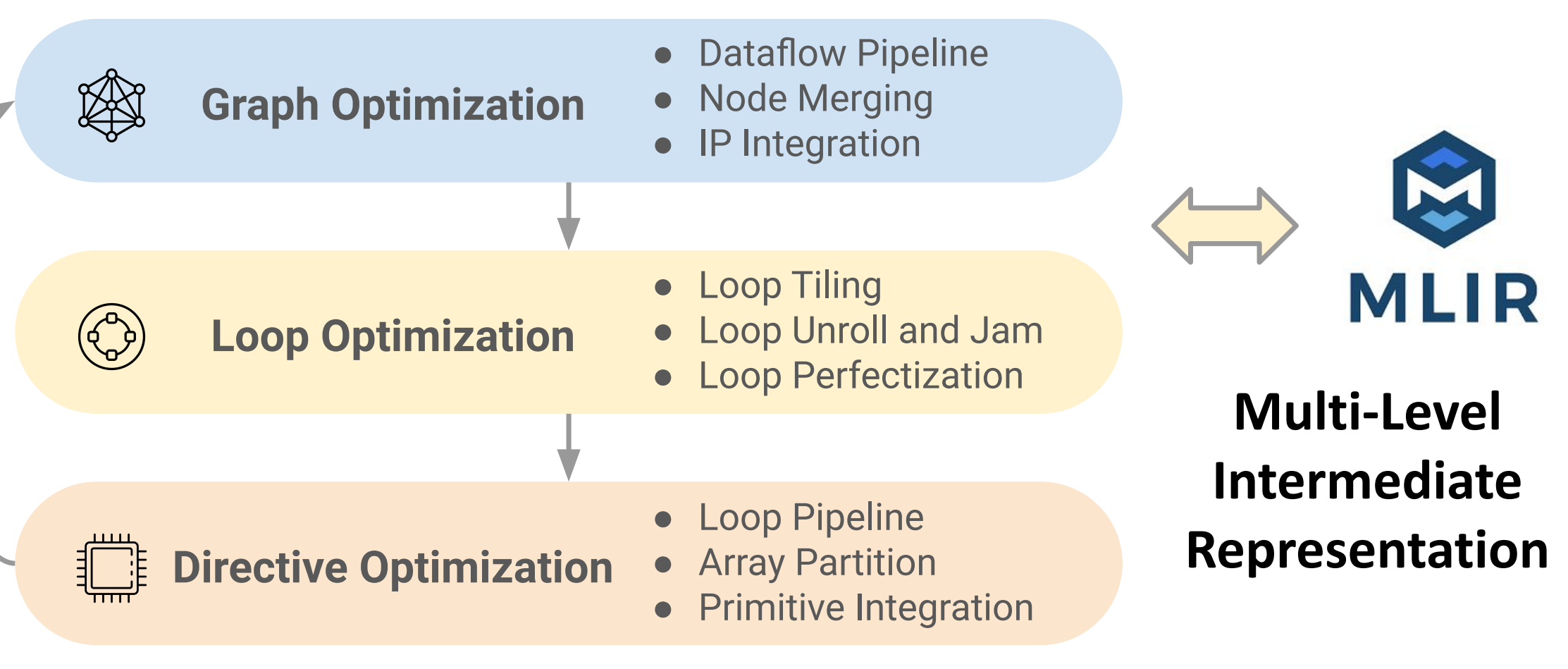
vHLS: Verifiable and Efficient High-Level Synthesis

Hanchen Ye, Deming Chen *UIUC (University of Illinois at Urbana-Champaign)*

Challenges and Motivation

HLS (High-Level Synthesis) has a great potential to continue to drive the high-productivity designs of circuits with high-density, high-energy efficiency, and short design cycle. However:

- Large-scale designs make it very challenging to comprehensively explore the large design space of different algorithmic choices and lead to sub-optimal design solutions -> **Efficiency**.
- Due to the complicated functionality and hardware hierarchy, verification properties are difficult to establish while the complexity of correctness proving restricts the scalability -> **Verification**.



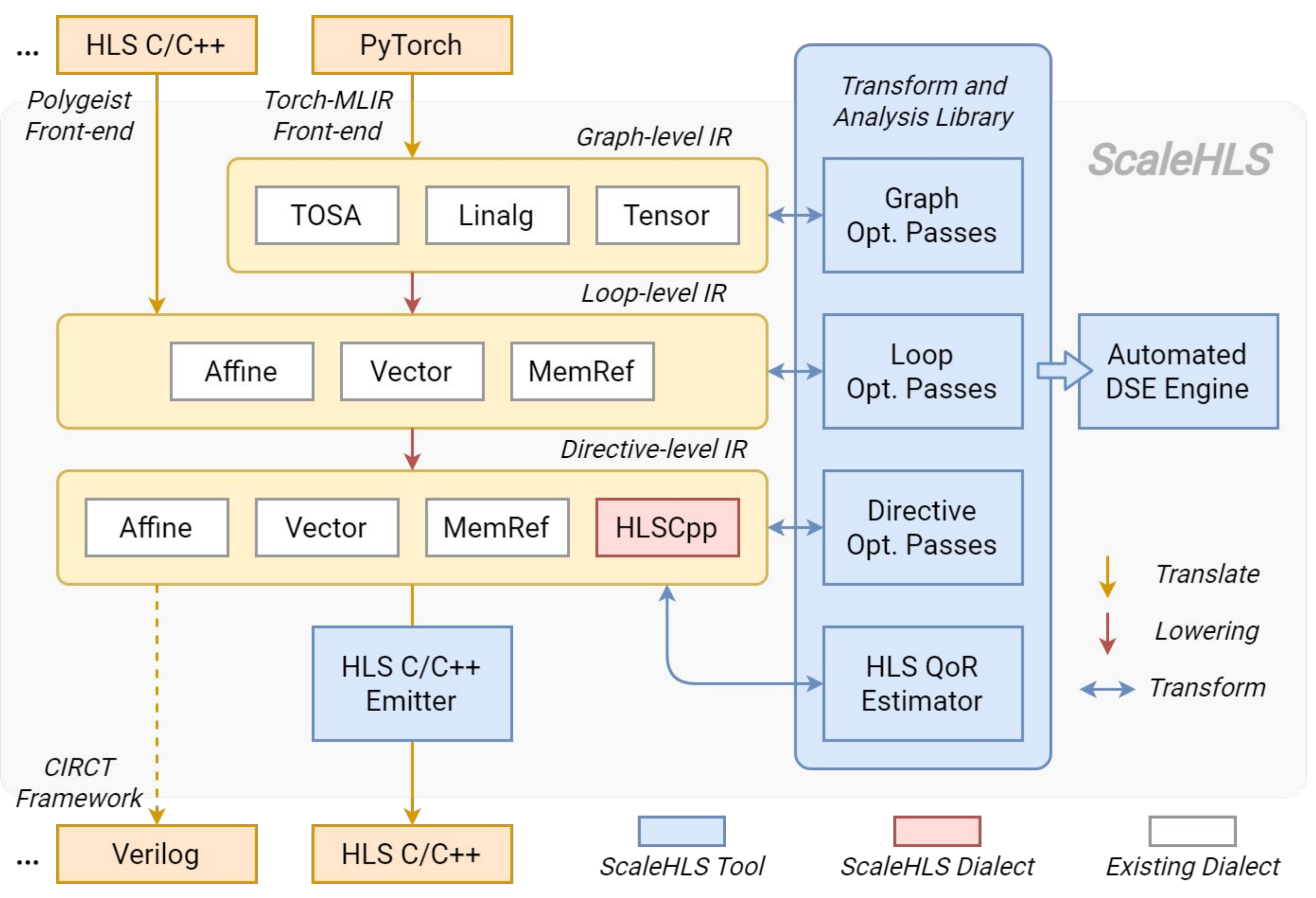
Marry HLS and MLIR

- Abstract HLS designs into multiple representation levels
- Solve the HLS optimization problems at "correct" abstraction levels
- Enable comprehensive design space exploration for optimal solutions
- Promote the verification and transform of HLS designs as first-class citizens

Leverage and Contribute to Large Community:

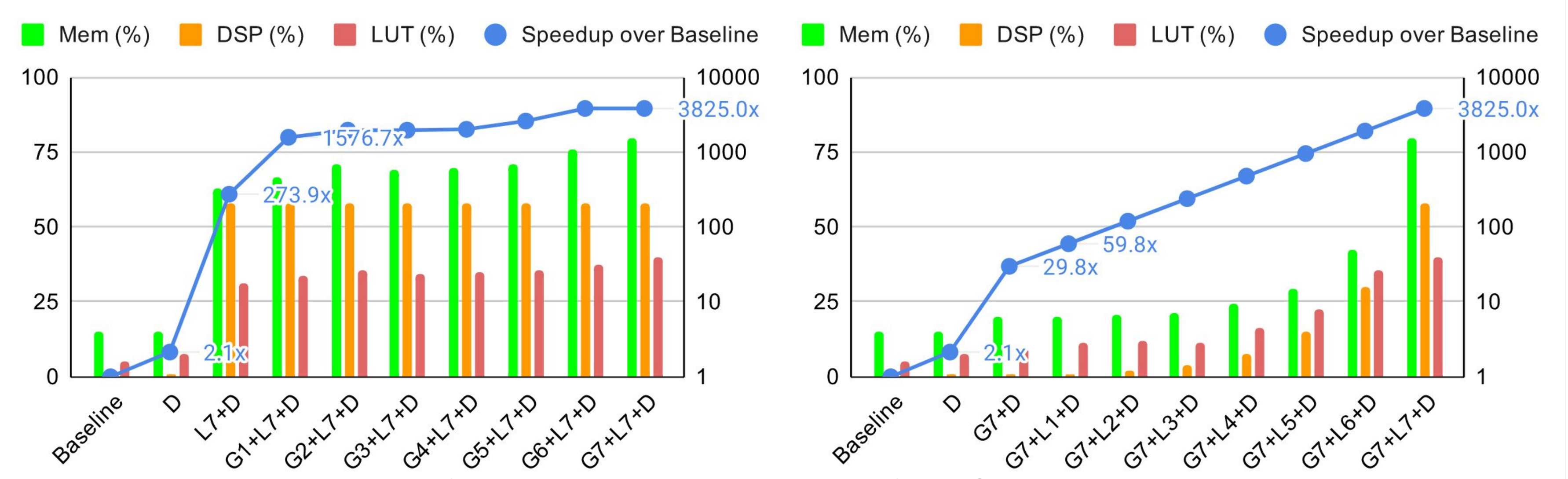
intel. NVIDIA. AMD SiFive Microsoft Google …….

ScaleHLS - HPCA'22, LATTE'21



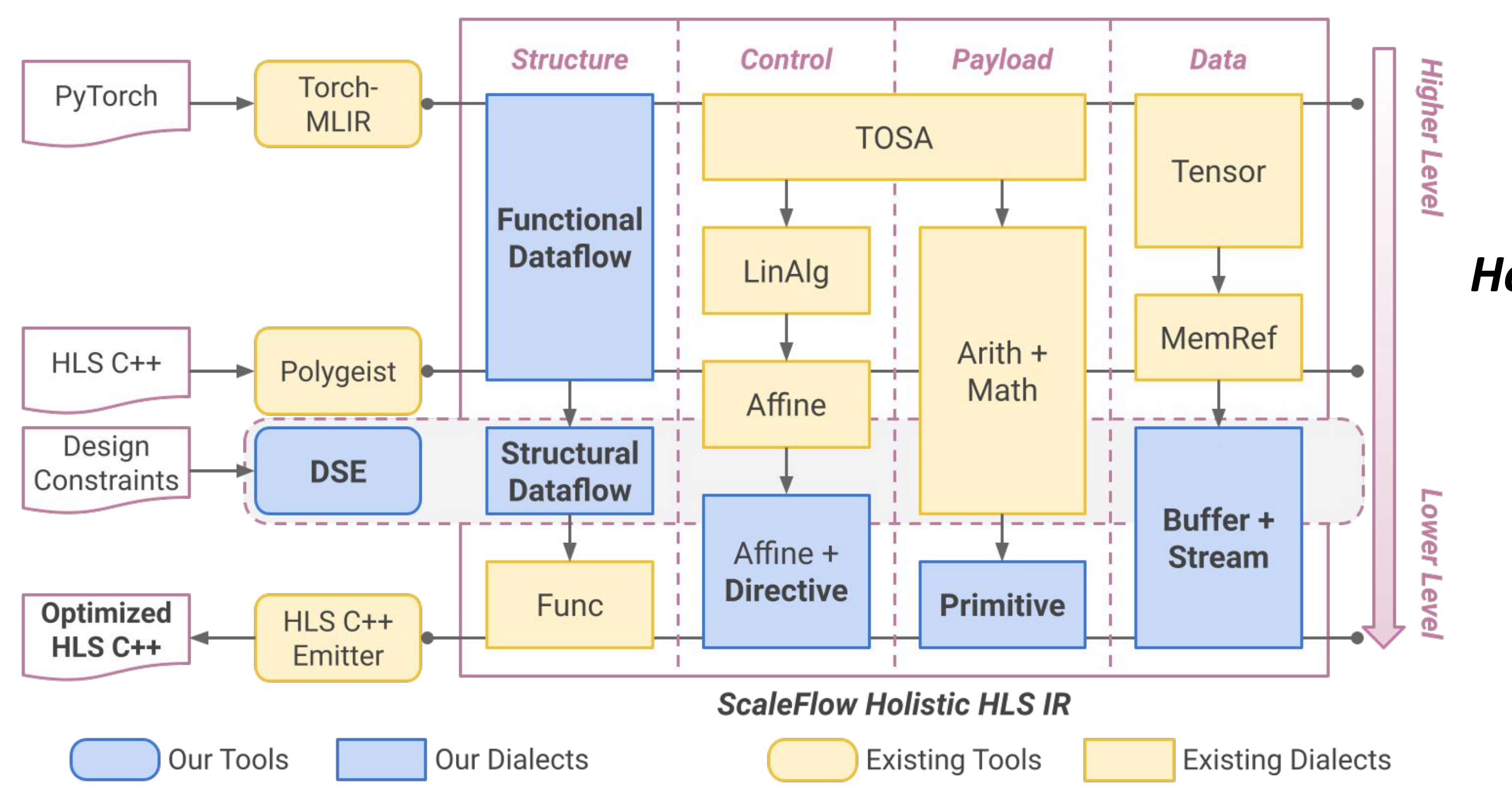
***First* Multi-Level HLS**
Representation
Optimization
Exploration

- **Fast** Compilation
- **Thorough** DSE
- **Efficient** U-Arch

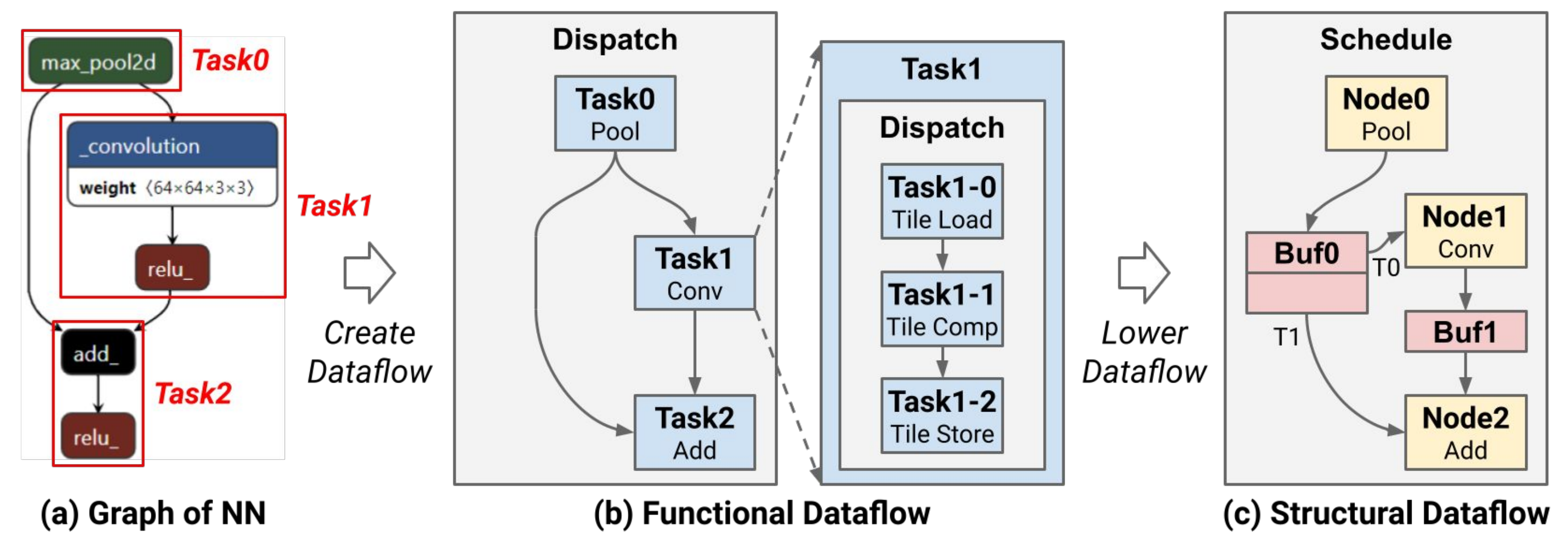


ScaleHLS Optimization Results of ResNet-18

ScaleFlow (WIP) - DAC'22, TRETS (under review)



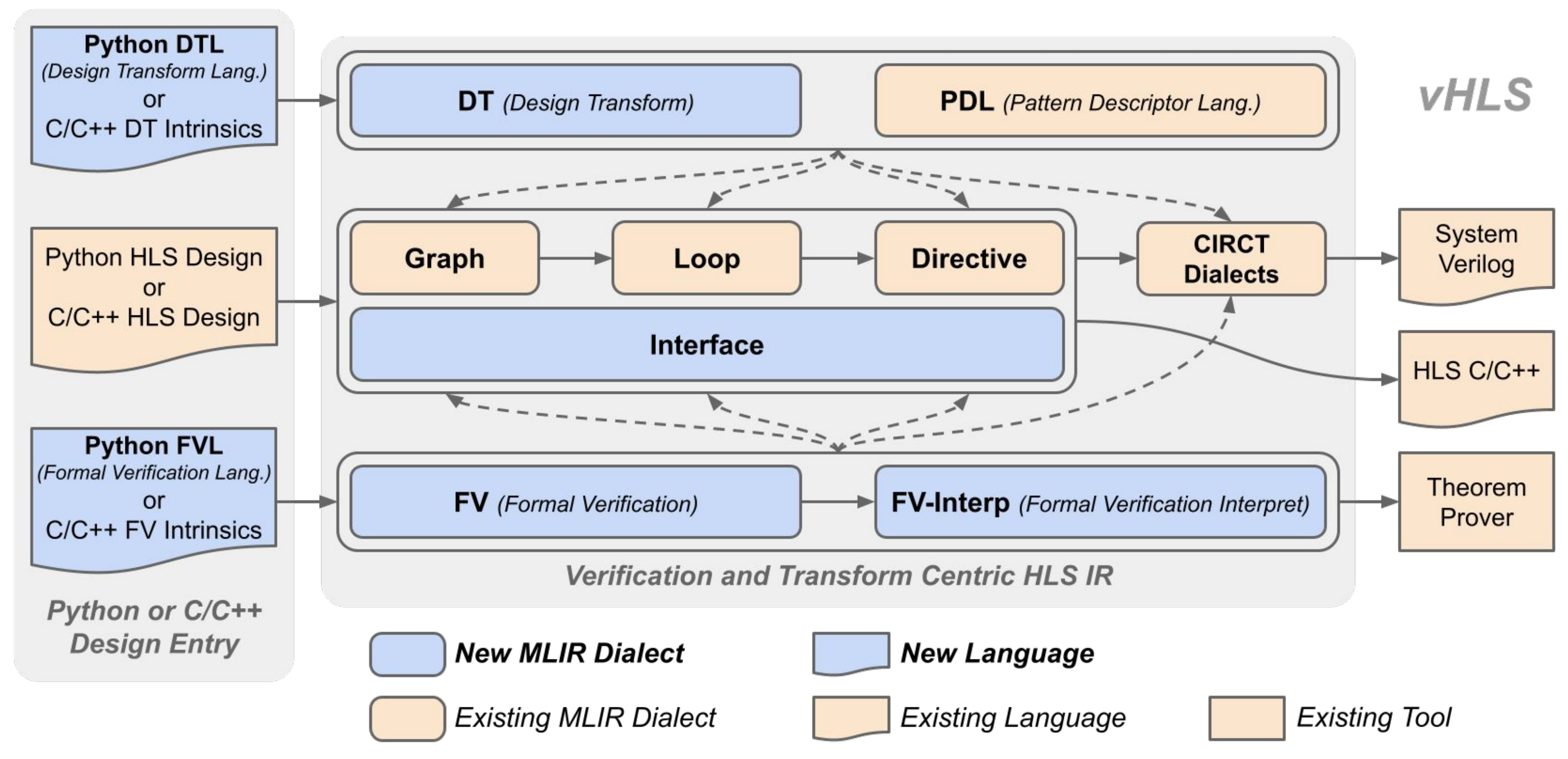
***First* Holistic HLS IR**
Structure
Control
Payload
Data



Two-Level Dataflow

- **Functional Dataflow**: High-level IR without hardware details for fast dataflow construction and task partition
- **Structural Dataflow:** Low-level IR with hardware details, such as the communication between dataflow nodes, for comprehensive dataflow scheduling, optimization, and design space exploration

vHLS (WIP)


Python DTL (Design Transform Lang.) or C/C++ DT Intrinsics
Python HLS Design or C/C++ HLS Design
Python FVL (Formal Verification Lang.) or C/C++ FV Intrinsics
Python or C/C++ Design Entry
DT (Design Transform)
PDL (Pattern Descriptor Lang.)
vHLS
Graph
Loop
Directive
CIRCT Dialects
Interface
FV (Formal Verification)
FV-Interp (Formal Verification Interpret)
Verification and Transform Centric HLS IR
System Verilog
HLS C/C++
Theorem Prover
New MLIR Dialect
New Language
Existing MLIR Dialect
Existing Language
Existing Tool

```
@fvl.method
def find_max_in_positive_seq(xs: seq[int]):
    fvl.requires(fvl.forall(x >= 0 for x in xs))
    fvl.requires(len(xs) > 0)
    ans: int = xs[0]
    for i in range(1, len(xs)):
        fvl.invariant(0 <= i <= len(xs))
        fvl.invariant(fvl.forall(
            xs[j] <= ans for j in range(0, i)))
        if ans < xs[i]:
            ans = xs[i]
    fvl.ensures(forall(x <= ans for x in xs))
  return ans
```

(a) FVL (Formal Verification Lang.)

```
func.func @find_max_in_positive_seq(
    %xs: memref<128xindex>) -> index {
  // fvl.require(fvl.forall(x >= 0 for x in xs))
  %c0 = arith.constant 0 : index
  %len = memref.dim %xs, %c0 : memref<128xindex>
  fv.require {
    %res = fv.for_all %x = %c0 to %len {
    %iter_res = arith.cmpi uge, %x, %c0 : index
    fv.yield %iter_res : i1
    }
    fv.yield %res : i1
  }
  // Rest of code...
}
```

(b) FV (Formal Verification) IR

Correct-by-Construction HLS
Design
Verification
Transform

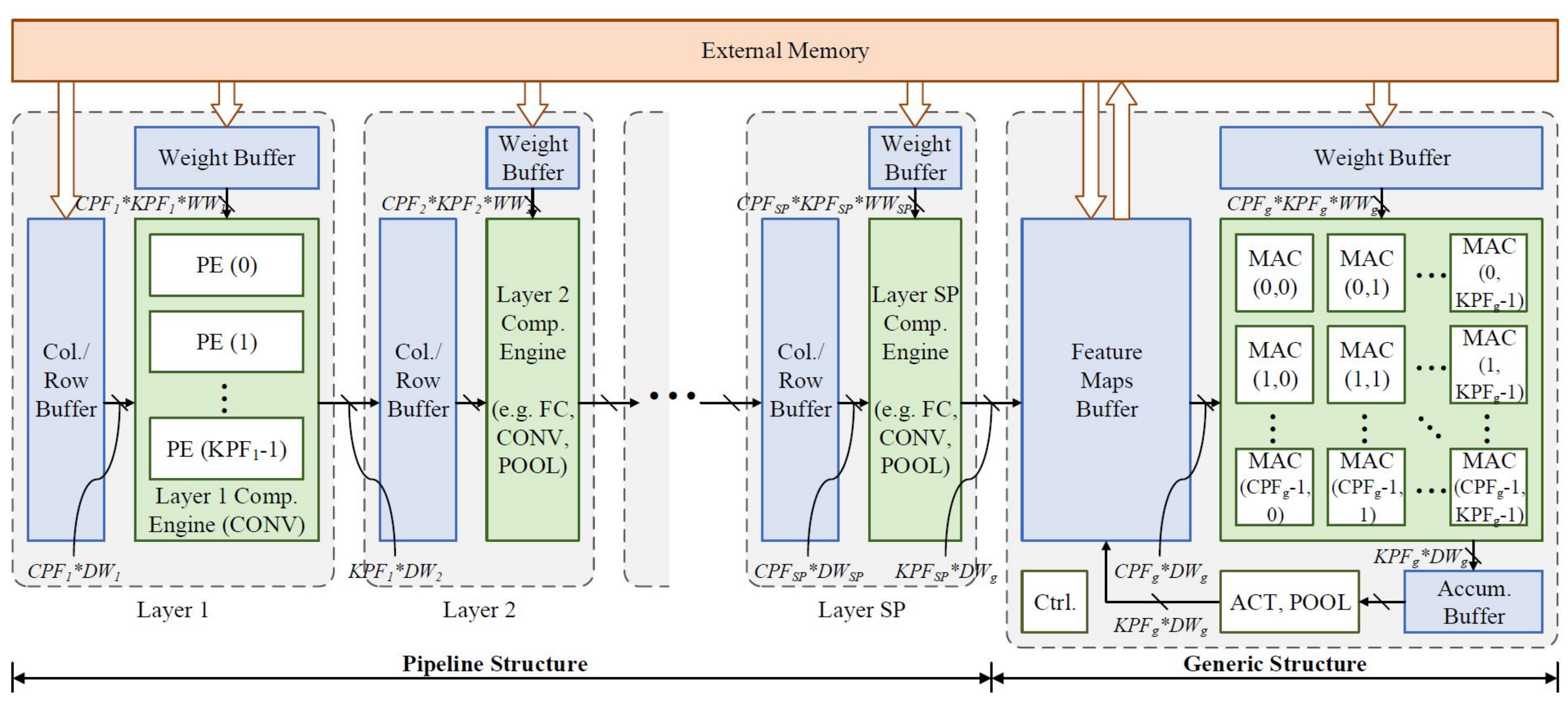
```
@dtl.is_pattern(benefit=0)
def pattern():
    a = dtl.value(dtl.Int(8))
    b = dtl.value(dtl.Int(8))
    c = dtl.value(dtl.Int(32))
    res = a * b + c
    loop_transform(res)
```

```
@dtl.is_transform
def loop_transform(res):
    loop = dtl.parent_loop(res)
    outer, inner = dtl.split(loop, 2)
    dtl.unroll(inner, 2)
    dtl.pipeline(outer)
```

(c) DTL (Design Transform Lang.)

- **FVL**: Formal verification language based on SMT theorem provers
- **DTL:** Design transform language based on hierarchical pattern matching and rewriting

HybridDNN - DAC'20, DNNExplorer - ICCAD'20



HLS-based Neural Network Acceleration
2.0x - 4.4x Speedup

- **Micro-Architecture**: Hybrid architecture that supports both Winograd and Spatial convolution; New accelerator paradigm combining pipeline structure and generic structure
- **Design Space Exploration:** Two-phase DSE with analysis-based local optimization and particle swarm-based global optimization

Open-Source Community

ScaleHLS GitHub Repository
https://github.com/hanchenye/scalehls
19,164 Views and **1,842** Downloads since Feb. 1, 2022

[1] **HPCA'22, H. Ye**, et al., *ScaleHLS: A New Scalable High-Level Synthesis Framework on Multi-Level Intermediate Representation*
[2] **LATTE'21, H. Ye**, et al., *ScaleHLS: Achieving Scalable High-Level Synthesis through MLIR*
[3] **TRETS** (under review), H. Jun, **H. Ye, et al.,** *AutoScaleDSE: A Scalable Design Space Exploration Engine for High-Level Synthesis*
[4] **DAC'22, H. Ye**, et al., *ScaleHLS: a Scalable High-Level Synthesis Framework with Multi-level Transformations and Optimizations*
[5] **DAC'20, H. Ye**, et al., *HybridDNN: A Framework for High-Performance Hybrid DNN Accelerator Design and Implementation*
[6] **ICCAD'20**, X. Zhang*, **H. Ye***, et al., *DNNExplorer: a framework for modeling and exploring a novel paradigm of FPGA-based DNN accelerator*