Paper accepted to HPCA 2022
No final paper released yet

ScaleHLS: Scalable High-Level Synthesis through MLIR

Hanchen Ye, Cong Hao, Jianyi Cheng, Hyunmin Jeong, Jack Huang, Stephen Neuendorffer, Deming Chen

Presenters: [Team 2] Insu Jang, Jae-Won Chung

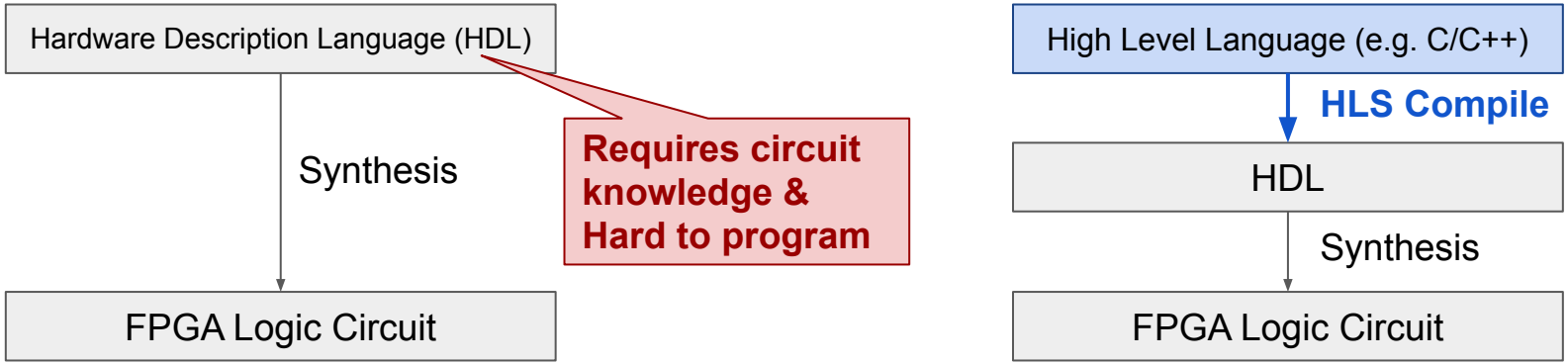
COLLEGE OF ENGINEERING
COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF MICHIGAN

Agenda

- Background
  - High-level synthesis (HLS)
  - Multi-level IR (MLIR)
- ScaleHLS
  - Goal and architecture
  - Dataflow optimization
  - Loop optimization
- Evaluation
- Conclusion

Background: High-level Synthesis

- Programming FPGA is hard, requiring circuit knowledge
- HLS: A way of implementing HW design with high level language

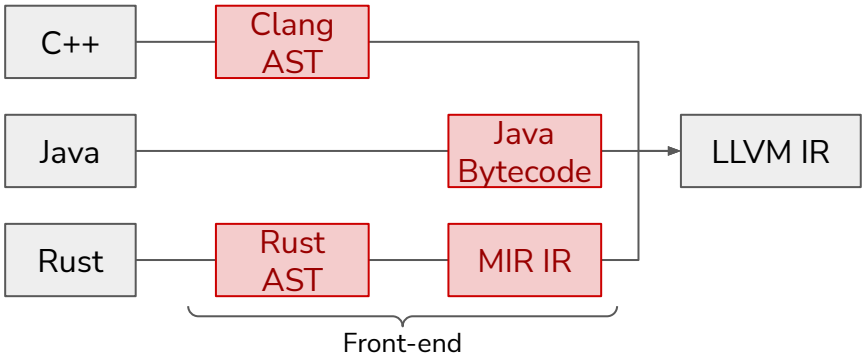


Background: Multi-level IR (MLIR)

- ScaleHLS built on top of MLIR

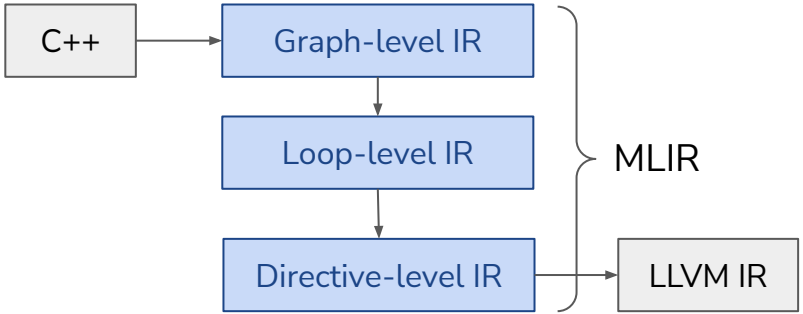
Before MLIR

- Each language implements its own high-level IR for its own optimization



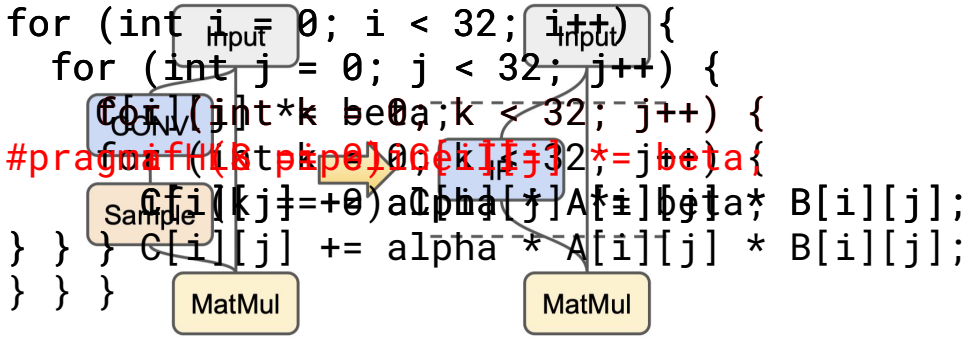
MLIR is

- easily customizable to add MLIR dialect or use common MLIR dialects



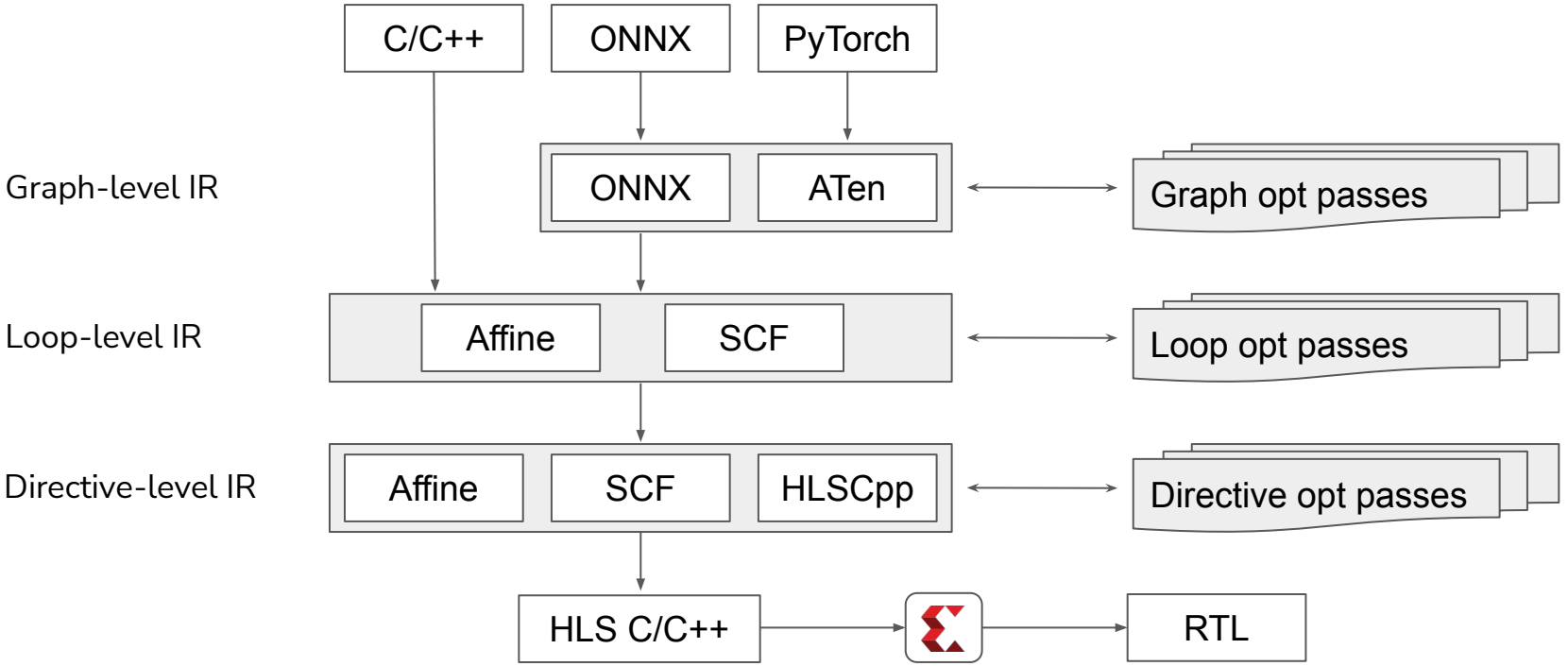
Goal of ScaleHLS

- End-to-end automation
- Capture different levels of optimization
  - Graph-level
  - Loop-level
  - Directive-level




```
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++) {
#pragma HLS pipeline II=1
      C[i][j] += alpha * A[i][k] * B[i][j];
} } } C[i][j] += alpha * A[i][j] * B[i][j];
} } }
```

ScaleHLS Architecture

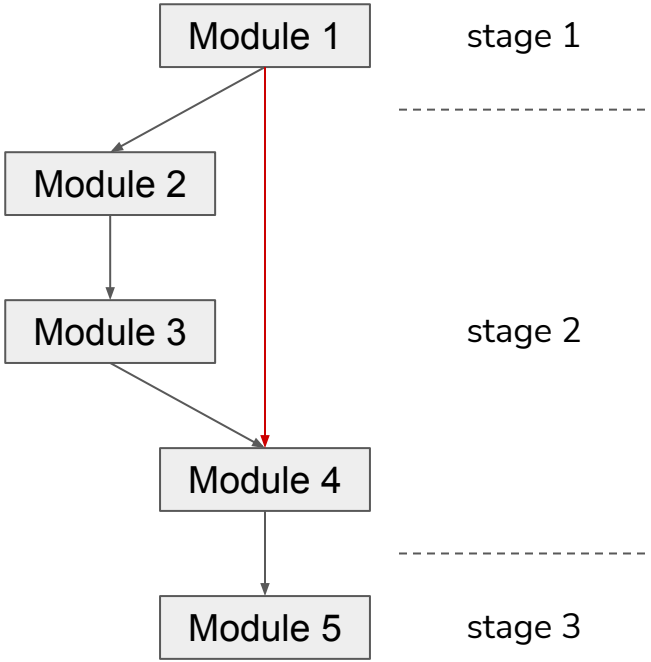


Optimizations

Level	Pass
Graph	Legalize dataflow
	Split function
Loop	Loop perfectization
	Loop order optimization
	Variable bound removal
	Loop tiling
	Loop unrolling

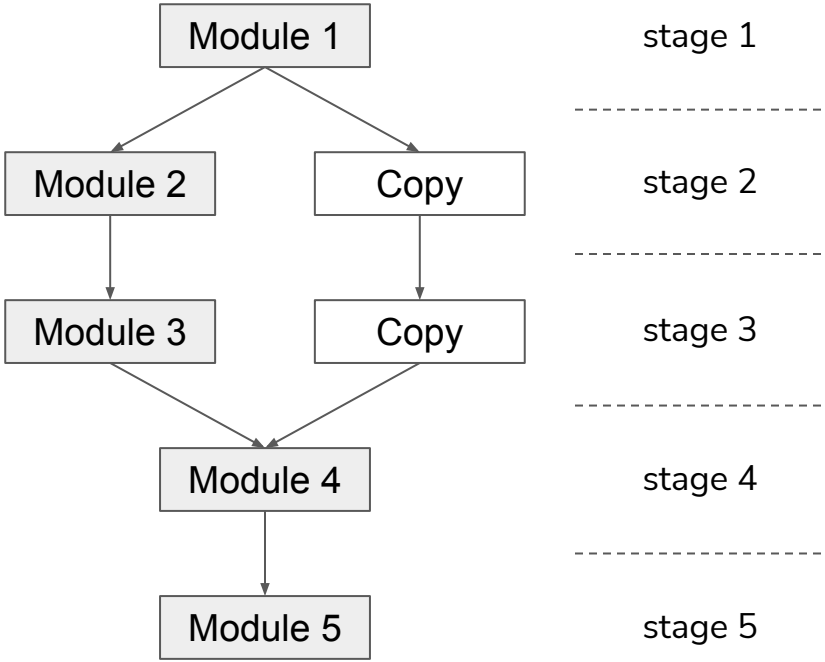
Level	Pass
Directive	Loop pipelining
	Function pipelining
	Array partitioning

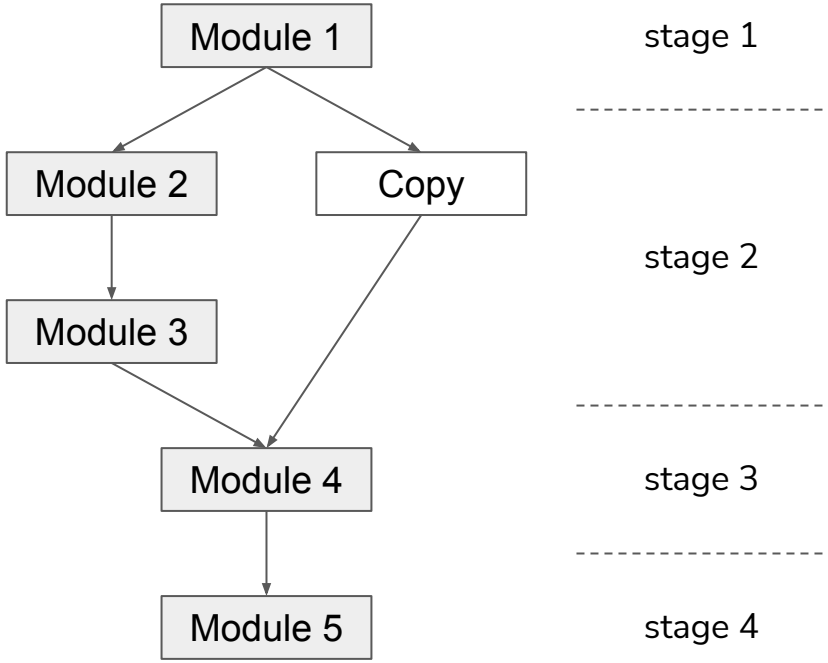
Graph-level Dataflow Optimization



A bypassing dataflow edge precludes finer-grained pipelining

Graph-level Dataflow Optimization





Loop Flattening and Perfectization

- **Loop flattening**: flatten multi-dimension nested loops into one loop

```
for (int i=0; i<16; i++) {
  for (int j=0; j<8; j++) {
    // ...
}}
```

```
for (int n=0; n<16*8; n+=8) {
  int i = n / 8;
  int j = n % 8;
  // ...
}
```

- Problem: **Imperfect loops** cannot be flattened
  - **Aut**o-level IR

Statements outside innermost loop → loops are imperfect

```
for (int i=0; i<16; i++) {
  a[i][0] *= n;
  for (int j=0; j<8; j++) {
    a[i][j] += m * b[i][j];
}}
```

Perfectization →

```
for (int i=0; i<16; i++) {
  for (int j=0; j<i; j++) {
    if (j==0) a[i][j] *= n;
    a[i][j] += m * b[i][j];
}}
```

Flattening →

```
for (int n=0; n<16*8; n+=8) {
  int i=n/8, j=n%8;
  if (j==0) a[i][j] *= n;
  a[i][j] += m * b[i][j];
}
```

Loop Unrolling and Pipelining

- Add a `pipeline` directives in d
  - Transformation is applied by Vivac
- Requirements: all sub-loops unrolled / sub-functions
  - **Legalize** the loop by unrolling sub-loop to be fully pipelined

- **k-loop is fully unrolled**
- **Pipelining is applied to `j`-loop**

*** `i-j` nested loops can be automatically flattened since this is perfect loop**

```
for (int i=0; i<16; i++) {
  for (int j=0; j<i; j++) {
    for (int k=0; k<8; k++) {
      a[i][j] += b[i][k] * b[j][k];
}}}
```

```
for (int i=0; i<16; i++) {
  for (int j=0; j<i; j++) {
    #pragma HLS pipeline II=3
    a[i][j] += b[i][0] * b[j][0];
    ...
    a[i][j] += b[i][7] * b[j][7];
}}
```

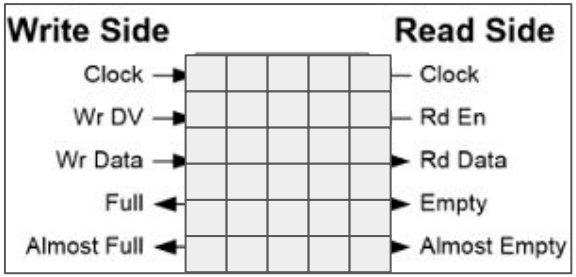
Array Partitioning

- Loop unroll may not be effective due to **limited FPGA memory ports**
  - BRAM in right can perform 1 read / 1 write in 1 cycle

```
for (int i=0; i<5; i++) {
    a[i][0] = ...;
    ...
    a[i][5] = ...;
}
```

6 cycles

Cannot be executed simultaneously



a[5][6]

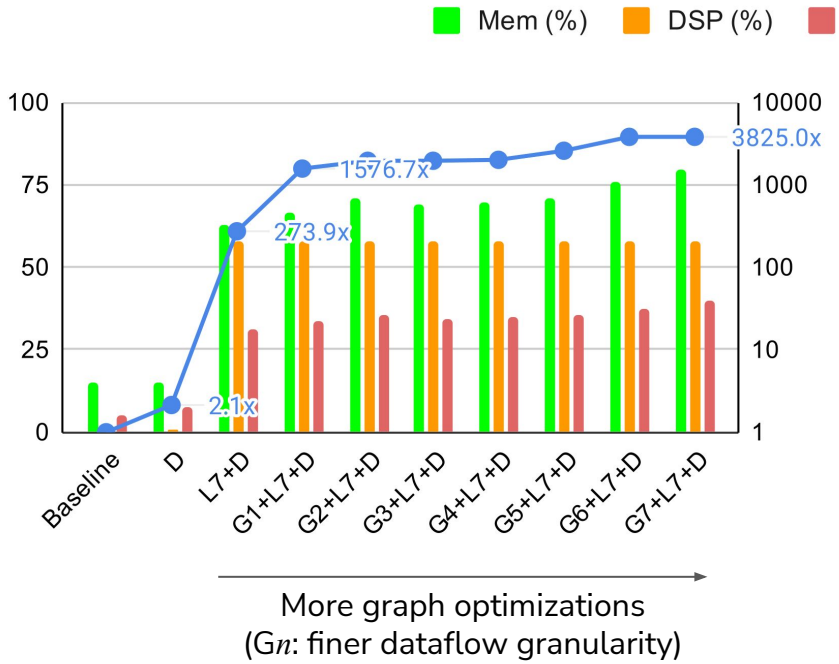
- **Automatically** detect memory access pattern and apply suitable array partition directives: `#pragma HLS array_partition variable=a ...`

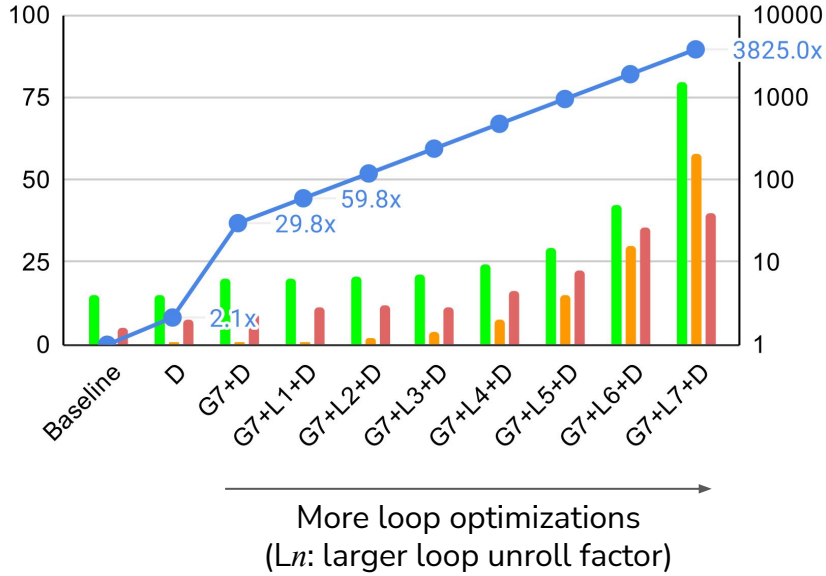
Evaluation: ResNet-18

- Translate PyTorch model definitions to Graph-level IR
- Results
  - **3.9x more resource efficient** versus TVM-VTA[1]
  - **3825.0x speedup** versus itself without any optimizations
  - **automatic optimization** versus nothing (This is the first work.)

[1] “A Hardware-Software Blueprint for Flexible Deep Learning Specialization,” Moreau et al., arXiv:1807.04188

Evaluation: ResNet-18





Conclusion and Commentary

- ScaleHLS: Optimizations for high performance HLS
  - Well-architected system with extensible design
  - Hardware-aware optimization passes and modelling
  - A lot of engineering effort invested
- Some limitations, though.
  - Weak evaluation baseline
  - Not many new things
  - Not sure why this is scalable

Paper accepted to HPCA 2022
No final paper released yet

ScaleHLS: Scalable High-Level Synthesis through MLIR

Questions? Comments?

Presenters: [Team 2] Insu Jang, Jae-Won Chung

COLLEGE OF ENGINEERING
COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF MICHIGAN