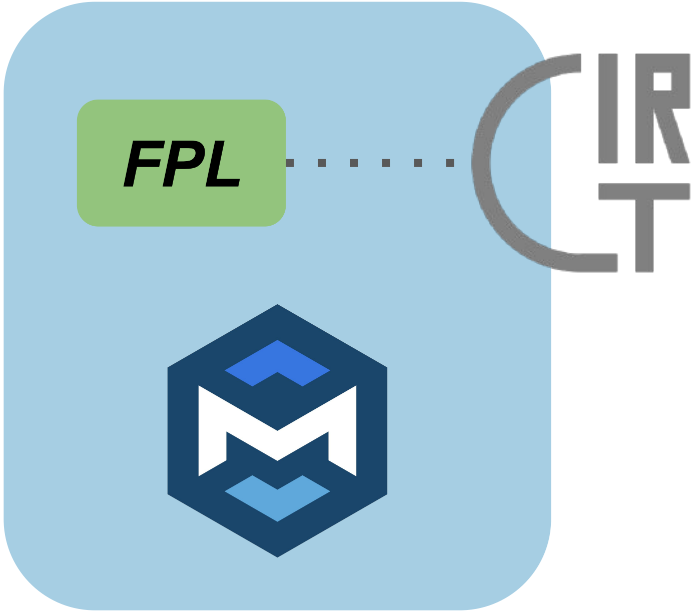
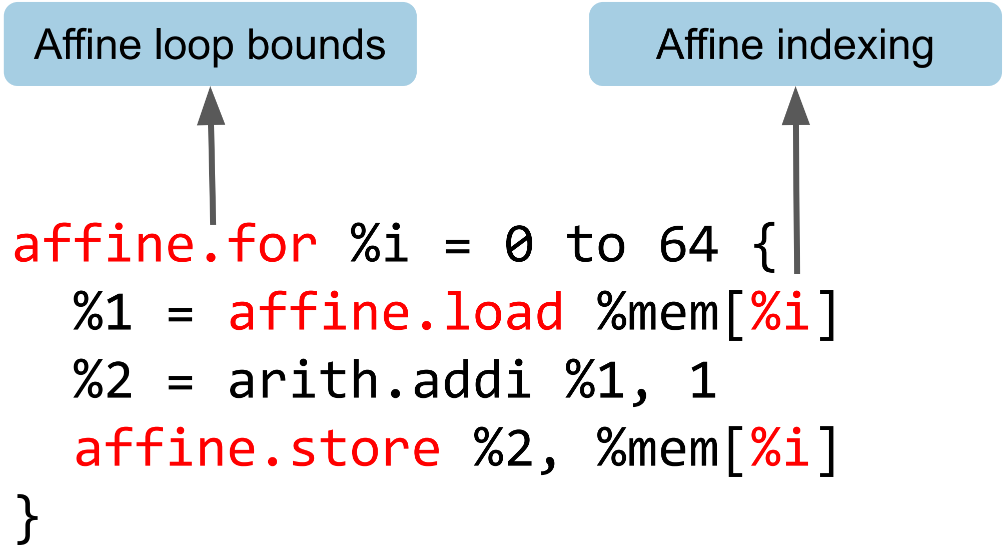
Building a Static HLS Pass with FPL

Kunwar Grover, Arjun Pitchanathan, Julian Oppermann, Mike Urbach, Tobias Grosser



IIIT Hyderabad, University of Edinburgh, TU Darmstadt, SiFive

Current Landscape

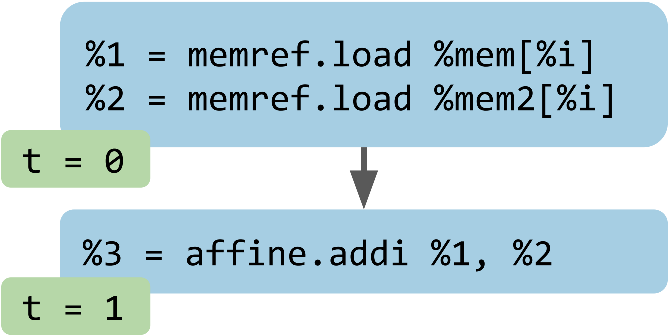


Affine loop bounds

Affine indexing

```
affine.for %i = 0 to 64 {
  %1 = affine.load %mem[%i]
  %2 = arith.addi %1, 1
  affine.store %2, %mem[%i]
}
```

Current Landscape



```
%1 = memref.load %mem[%i]
%2 = memref.load %mem2[%i]
```

t = 0

```
%3 = affine.addi %1, %2
```

t = 1

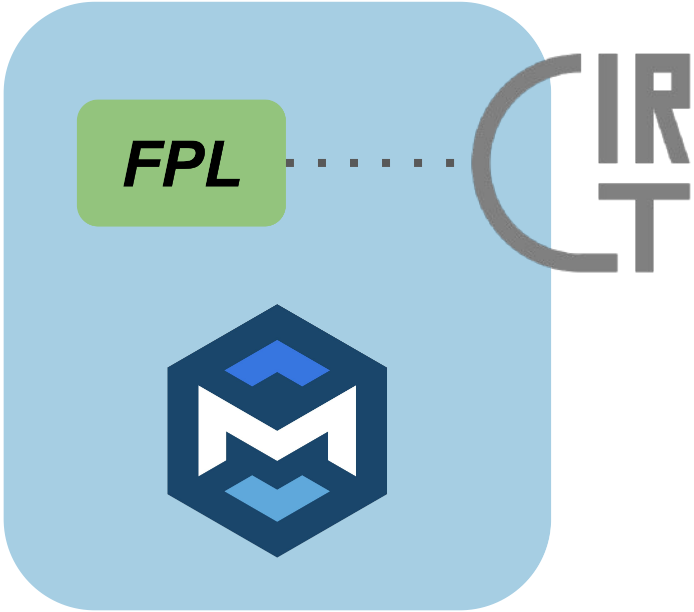
Current Landscape

FPL

Bringing the communities a little closer

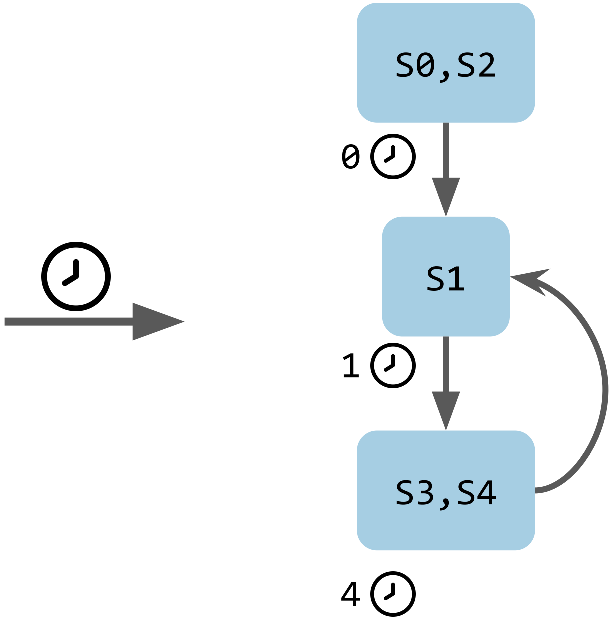
FPL

Bringing the communities a little closer



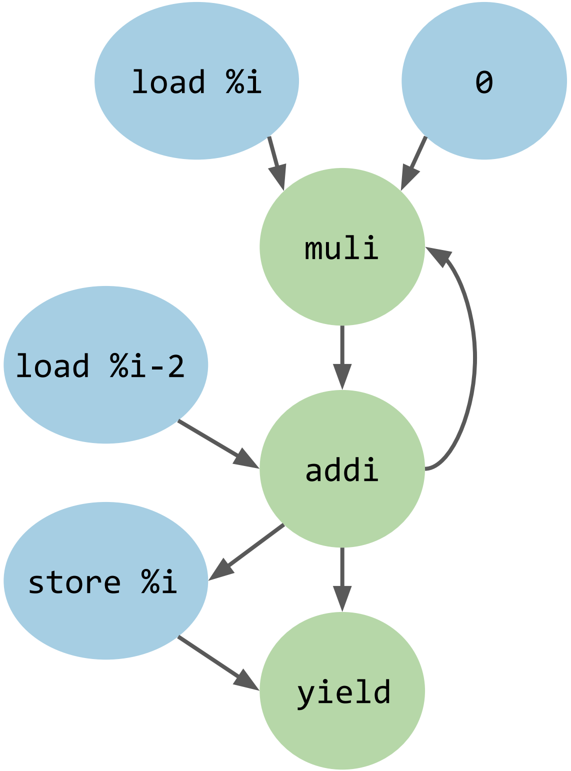
Static HLS in CIRCT

```
affine.for %i = 2 to 64 iter_args(%arg3 = 0) {
  S0: %1 = affine.load %mem[%i]
  S1: %3 = arith.muli %1, %arg3
  S2: %2 = affine.load %mem[%i - 2]
  S3: %4 = arith.addi %3, %2
  S4: affine.store %4, %mem[%i]
  S5: affine.yield %4
}
```



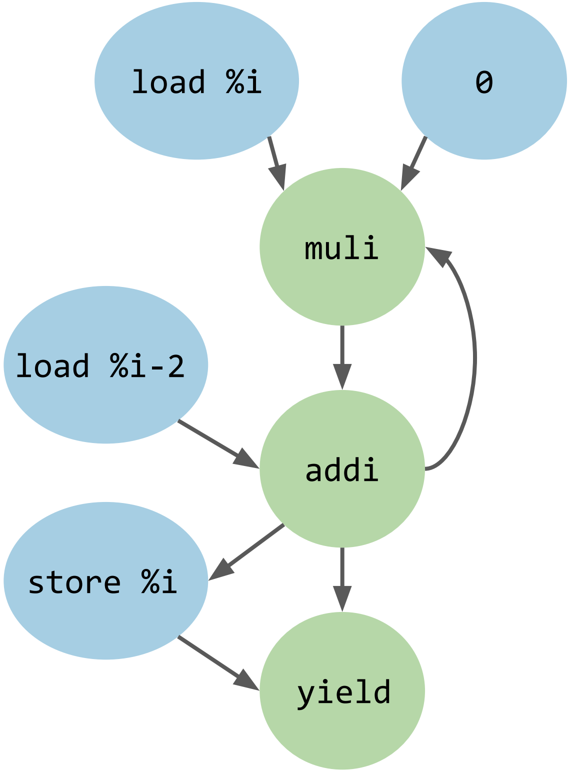
Building a model

```
affine.for %i = 2 to 64 iter_args(%arg3 = 0) {
  %1 = affine.load %mem[%i]
  %3 = arith.muli %1, %arg3
  %2 = affine.load %mem[%i - 2]
  %4 = arith.addi %3, %2
  affine.store %4, %mem[%i]
  affine.yield %4
}
```



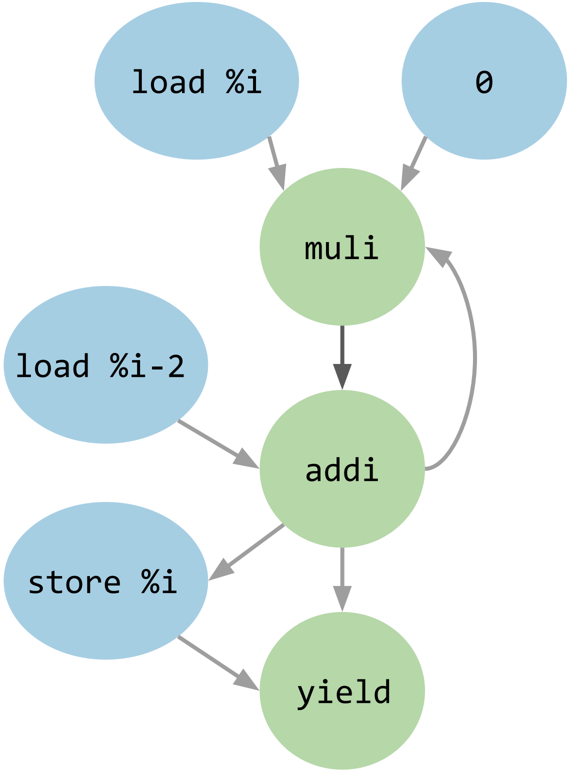
Building a model: Structural Dependencies

```
affine.for %i = 2 to 64 iter_args(%arg3 = 0) {
  %1 = affine.load %mem[%i]
  %3 = arith.muli %1, %arg3
  %2 = affine.load %mem[%i - 2]
  %4 = arith.addi %3, %2
  affine.store %4, %mem[%i]
  affine.yield %4
}
```



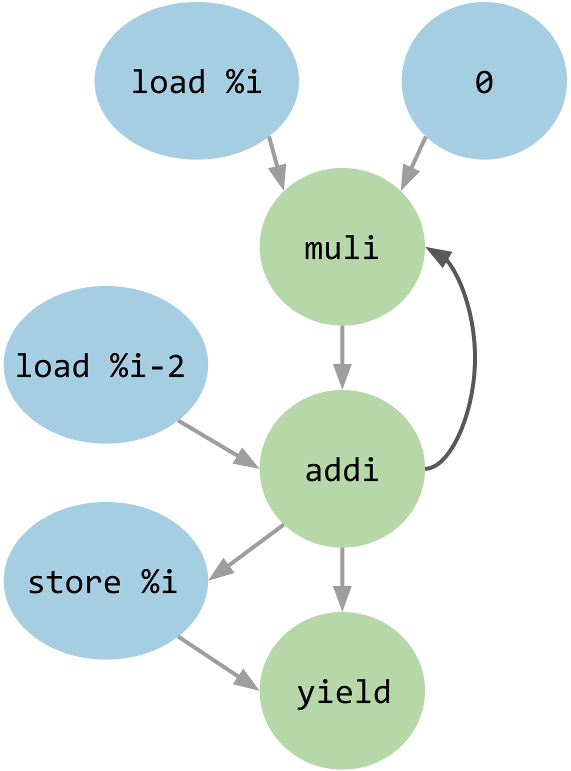
Building a model: Structural Dependencies

```
affine.for %i = 2 to 64 iter_args(%arg3 = 0) {
  %1 = affine.load %mem[%i]
  %3 = arith.muli %1, %arg3
  %2 = affine.load %mem[%i - 2]
  %4 = arith.addi %3, %2
  affine.store %4, %mem[%i]
  affine.yield %4
}
```



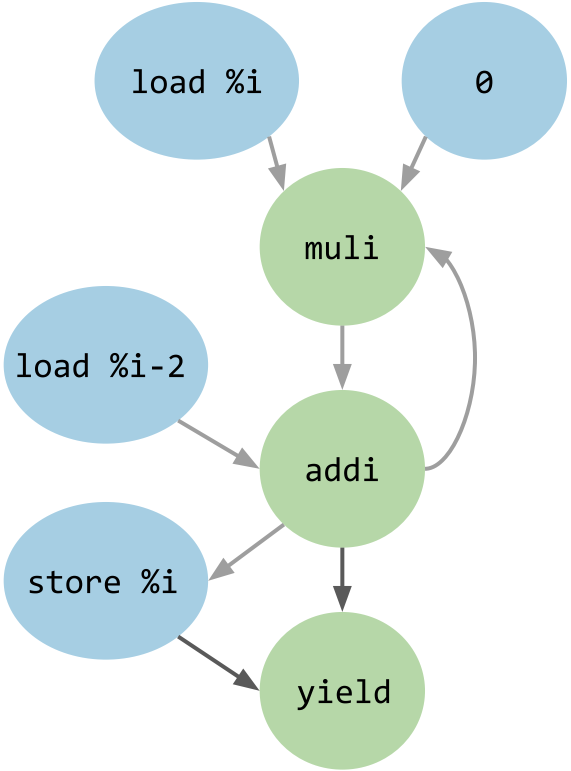
Building a model: Structural Dependencies

```
affine.for %i = 2 to 64 iter_args(%arg3 = 0) {
  %1 = affine.load %mem[%i]
  %3 = arith.muli %1, %arg3
  %2 = affine.load %mem[%i - 2]
  %4 = arith.addi %3, %2
  affine.store %4, %mem[%i]
  affine.yield %4
}
```



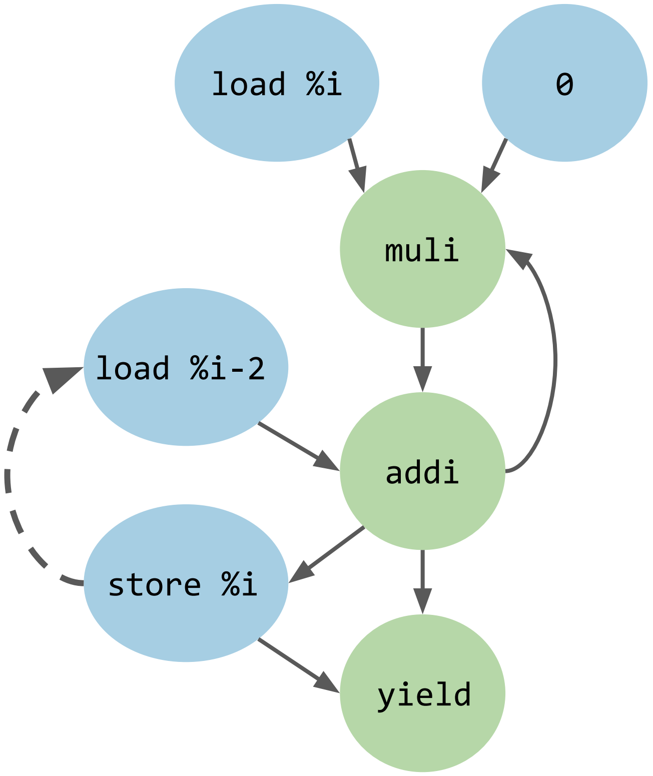
Building a model: Structural Dependencies

```
affine.for %i = 2 to 64 iter_args(%arg3 = 0) {
  %1 = affine.load %mem[%i]
  %3 = arith.muli %1, %arg3
  %2 = affine.load %mem[%i - 2]
  %4 = arith.addi %3, %2
  affine.store %4, %mem[%i]
  affine.yield %4
}
```



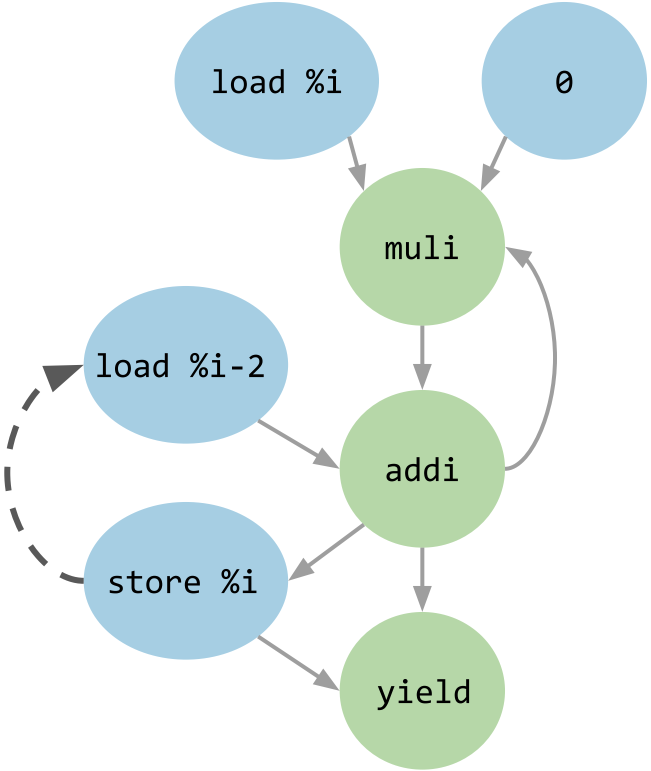
Building a model: Structural + Memory Dependencies

```
affine.for %i = 2 to 64 iter_args(%arg3 = 0) {
  %1 = affine.load %mem[%i]
  %3 = arith.muli %1, %arg3
  %2 = affine.load %mem[%i - 2]
  %4 = arith.addi %3, %2
  affine.store %4, %mem[%i]
  affine.yield %4
}
```



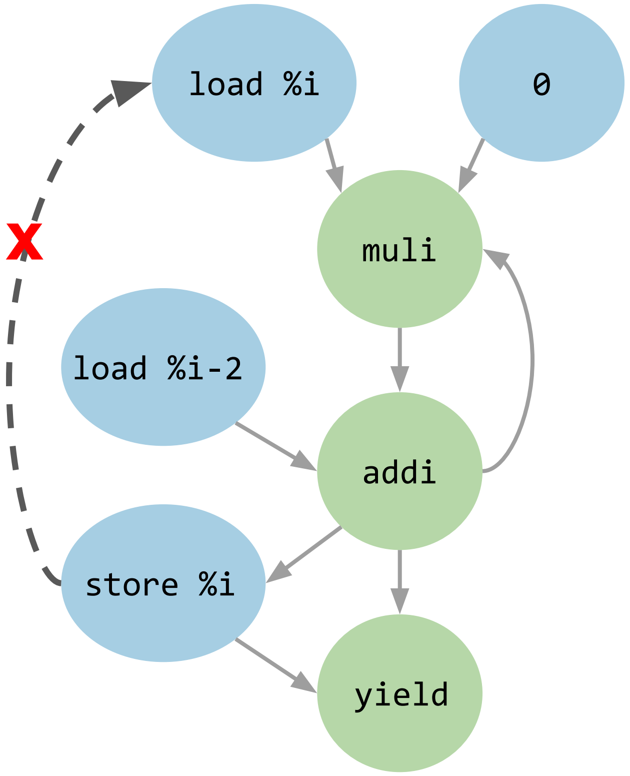
Building a model: Structural + Memory Dependencies

```
affine.for %i = 2 to 64 iter_args(%arg3 = 0) {
  %1 = affine.load %mem[%i]
  %3 = arith.muli %1, %arg3
  %2 = affine.load %mem[%i - 2]
  %4 = arith.addi %3, %2
  affine.store %4, %mem[%i]
  affine.yield %4
}
```



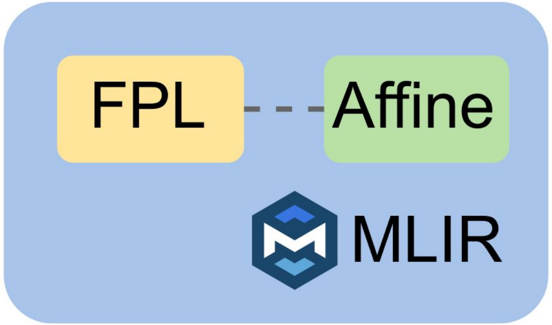
Building a model: Structural + Memory Dependencies

```
affine.for %i = 2 to 64 iter_args(%arg3 = 0) {
  %1 = affine.load %mem[%i]
  %3 = arith.muli %1, %arg3
  %2 = affine.load %mem[%i - 2]
  %4 = arith.addi %3, %2
  affine.store %4, %mem[%i]
  affine.yield %4
}
```

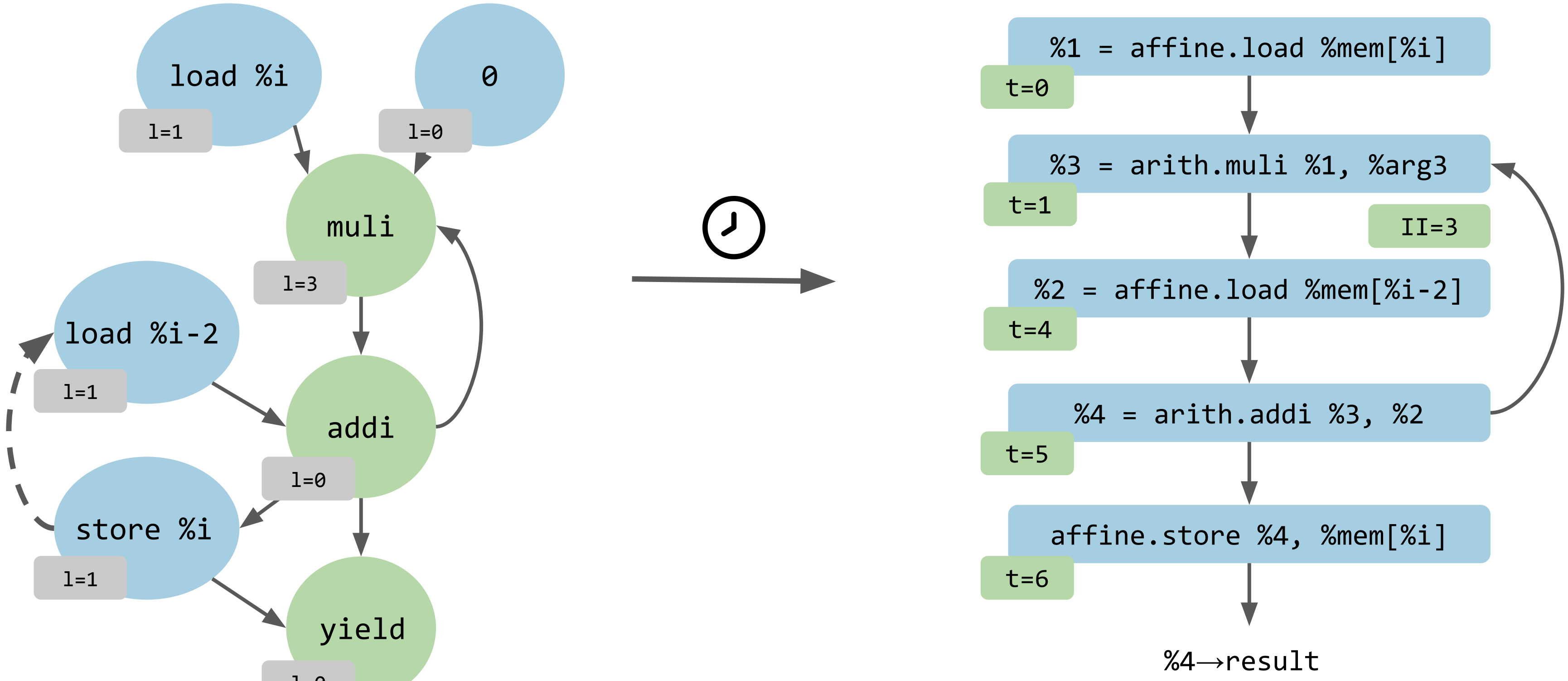


Polyhedral Dependence Analysis

```
affine.for %i = 2 to 64 iter_args(%arg3 = 0) {
  %1 = affine.load %mem[%i]
  %3 = arith.muli %1, %arg3
  %2 = affine.load %mem[%i - 2]
  %4 = arith.addi %3, %2
  affine.store %4, %mem[%i]
  affine.yield %4
}
```

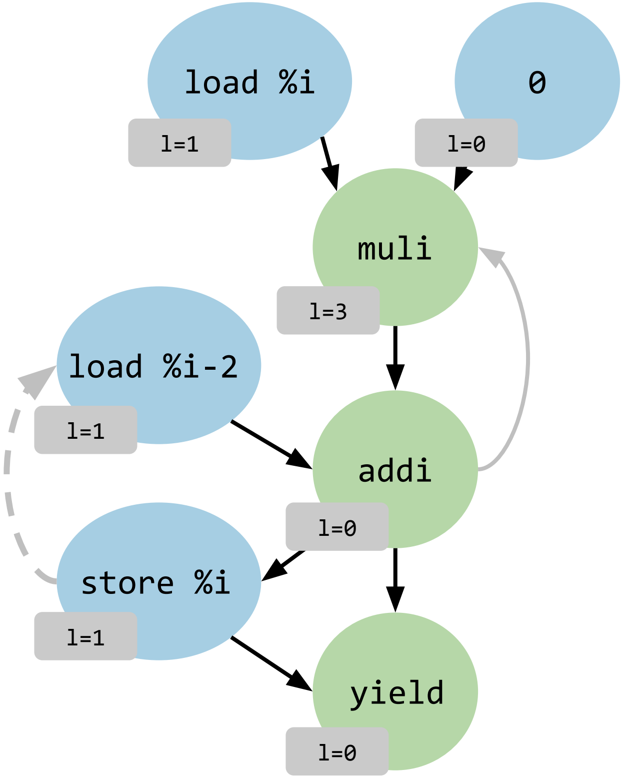


Scheduling the Problem



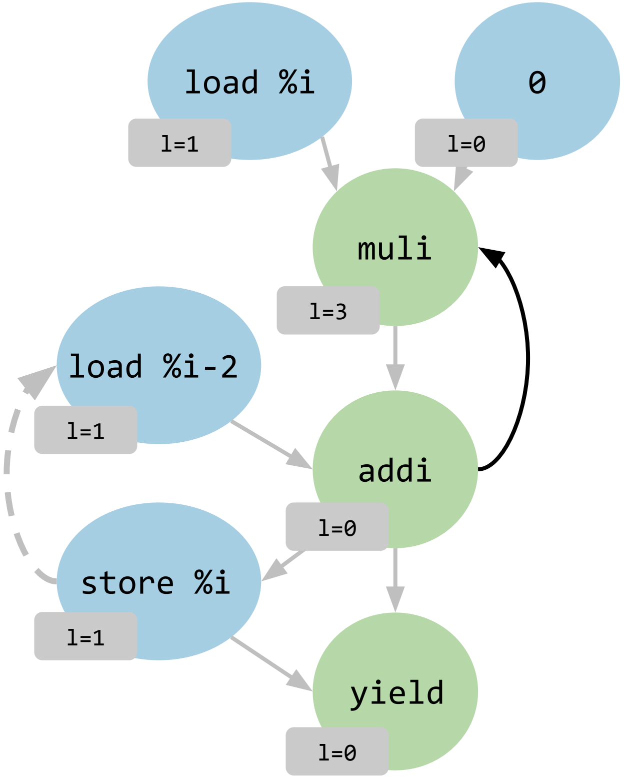
```
%1 = affine.load %mem[%i]
%3 = arith.muli %1, %arg3
%2 = affine.load %mem[%i-2]
%4 = arith.addi %3, %2
affine.store %4, %mem[%i]
%4→result
```

FPL for scheduling



```
depTime >= srcTime + srcLatency
```

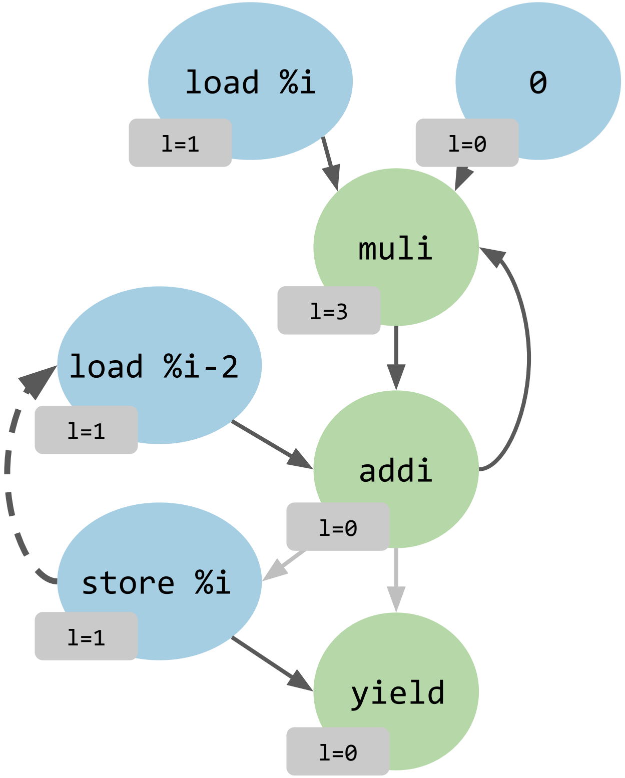
FPL for scheduling



```
depTime >= srcTime + srcLatency
```

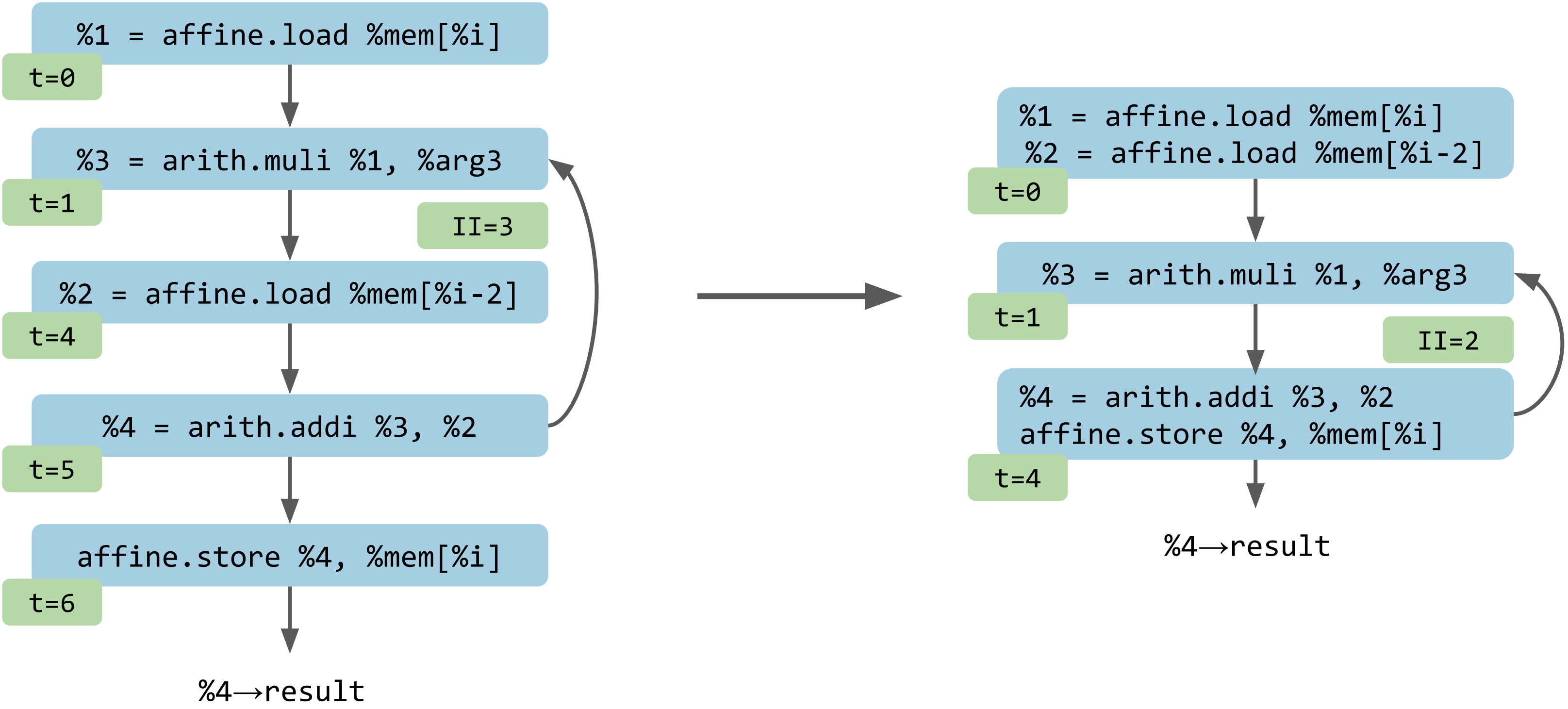
```
depTime >= srcTime + II + srcLatency
```

FPL for scheduling



`(II, objective, startTimes)`

Static HLS in CIRCT: Output Pipeline



%1 = affine.load %mem[%i]

t=0

%3 = arith.muli %1, %arg3

t=1

II=3

%2 = affine.load %mem[%i-2]

t=4

%4 = arith.addi %3, %2

t=5

affine.store %4, %mem[%i]

t=6

%4→result

%1 = affine.load %mem[%i]
%2 = affine.load %mem[%i-2]

t=0

%3 = arith.muli %1, %arg3

t=1

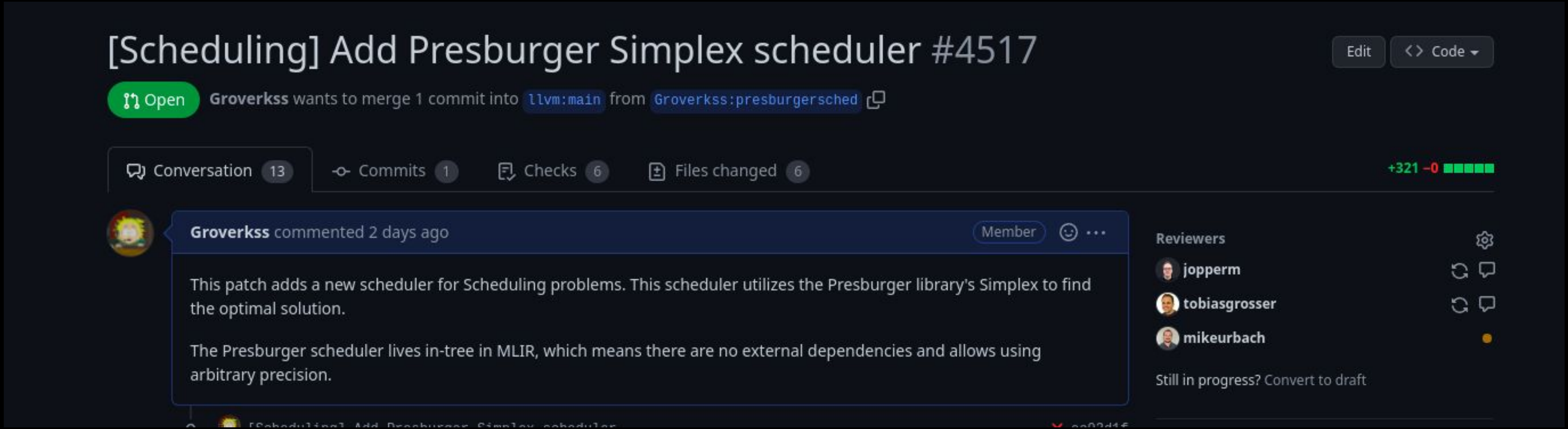
II=2

%4 = arith.addi %3, %2
affine.store %4, %mem[%i]

t=4

%4→result

We contribute to the CIRCT open-source project



[Scheduling] Add Presburger Simplex scheduler #4517

Open **Groverkss** wants to merge 1 commit into `llvm:main` from `Groverkss:presburgersched`

Groverkss commented 2 days ago Member

This patch adds a new scheduler for Scheduling problems. This scheduler utilizes the Presburger library's Simplex to find the optimal solution.

The Presburger scheduler lives in-tree in MLIR, which means there are no external dependencies and allows using arbitrary precision.

`[Scheduling] Add Presburger Simplex scheduler` ✕ `ee02d1f`

Bringing the communities together with FPL