Alexandru Calotoiu, Tal Ben-Nun, Noah Hütter, Paul Scheffler, Torsten Hoefler, Luca Benini

DaFlEx: Performance portability through dataflow extraction

Outline

- Introduction to DaCe
- DaFlEx overview
- DaFlEx first steps and goals

DaCe Overview

Domain Scientist

Problem Formulation

$$\frac{\partial u}{\partial t} - \alpha \nabla^2 u = 0$$

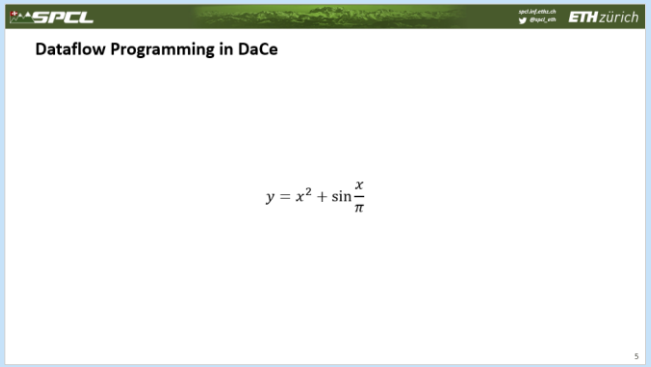
Python | DSLs

TensorFlow | MATLAB

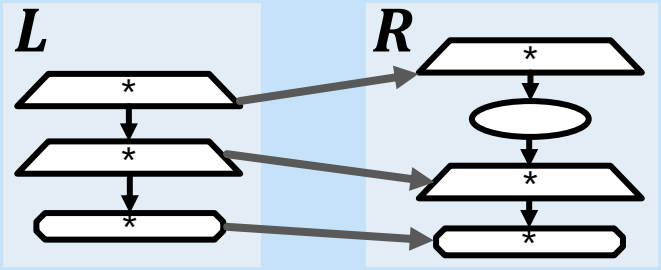
...

Scientific Frontend

Performance Engineer



Data-Centric Intermediate Representation (SDFG)



Graph Transformations

Transformed Dataflow

Performance Results

System

Hardware Information

Compiler

Runtime

CPU Binary

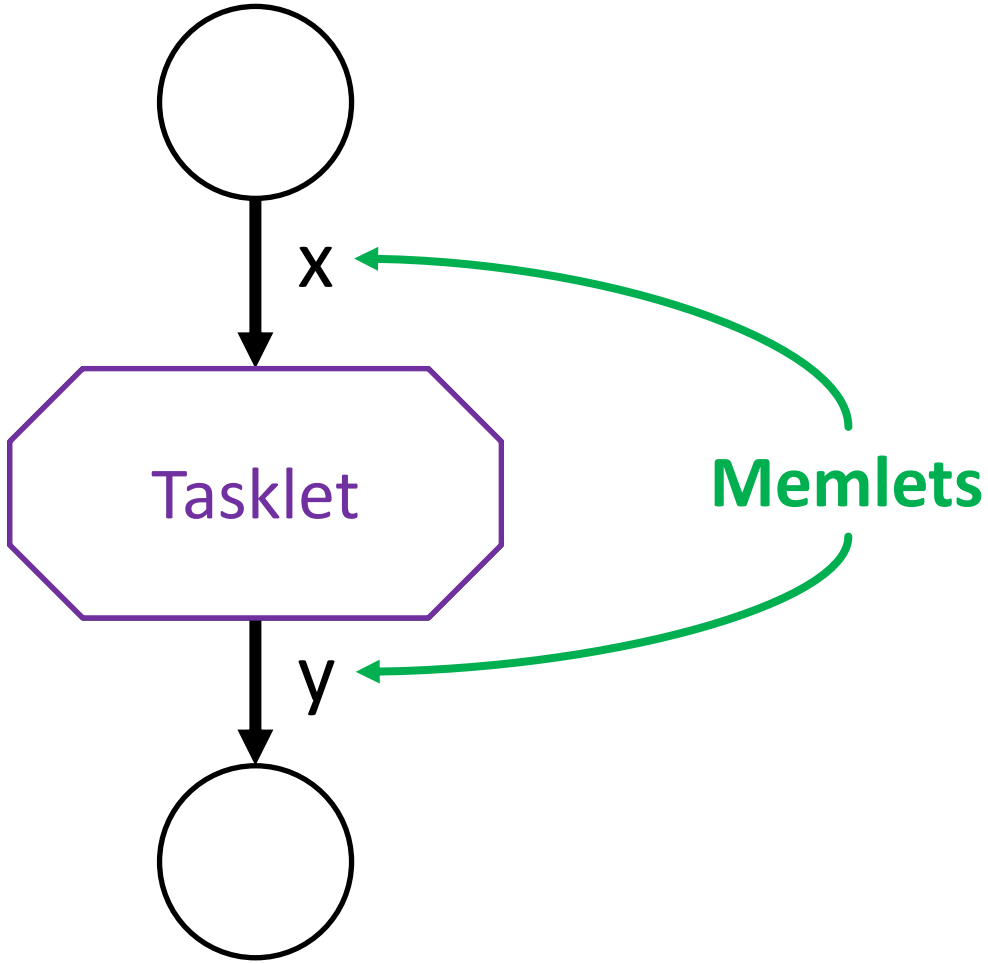
GPU Binary

FPGA Modules

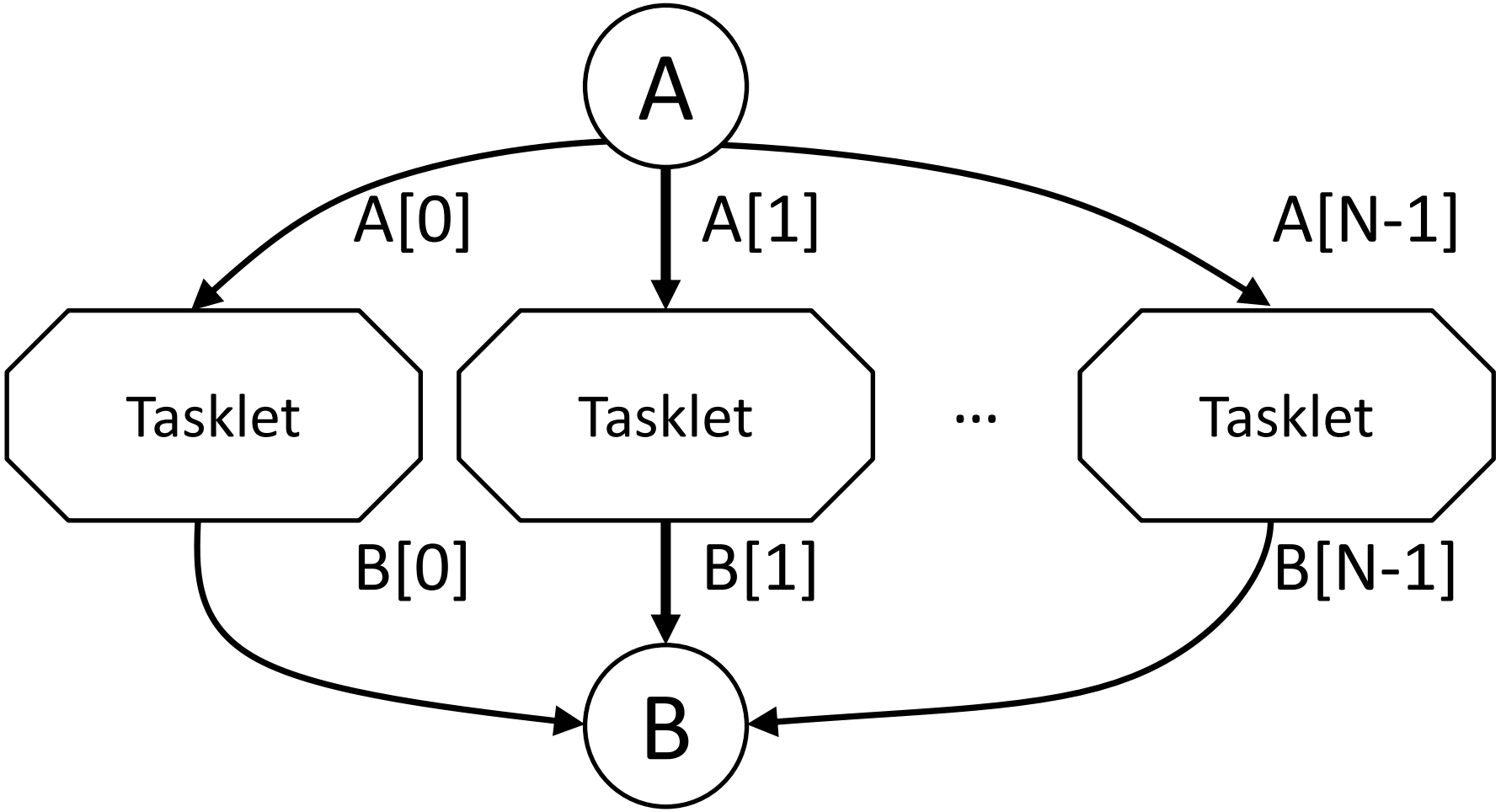
Dataflow Programming in DaCe

$$y = x^2 + \sin\frac{x}{\pi}$$

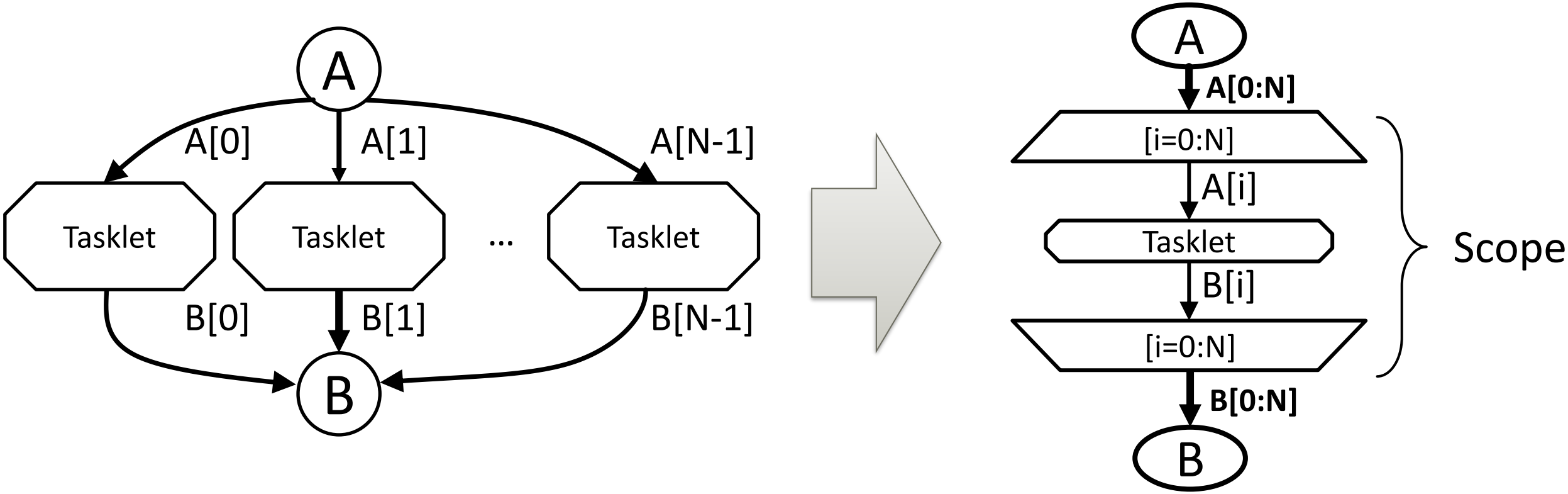
Dataflow Programming in DaCe



Parallel Dataflow Programming



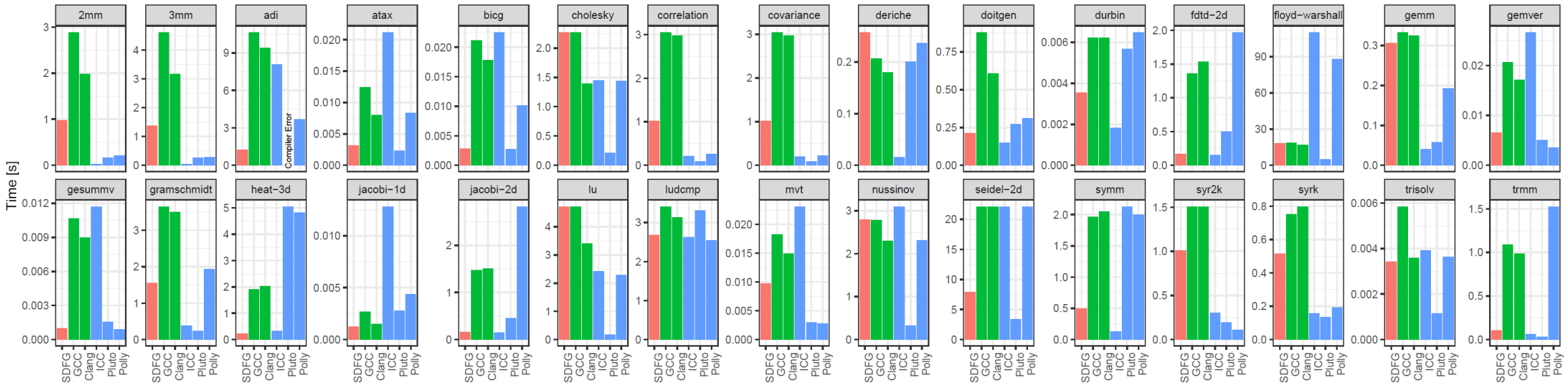
Parallel Dataflow Programming



Performance Evaluation: Polybench (CPU)

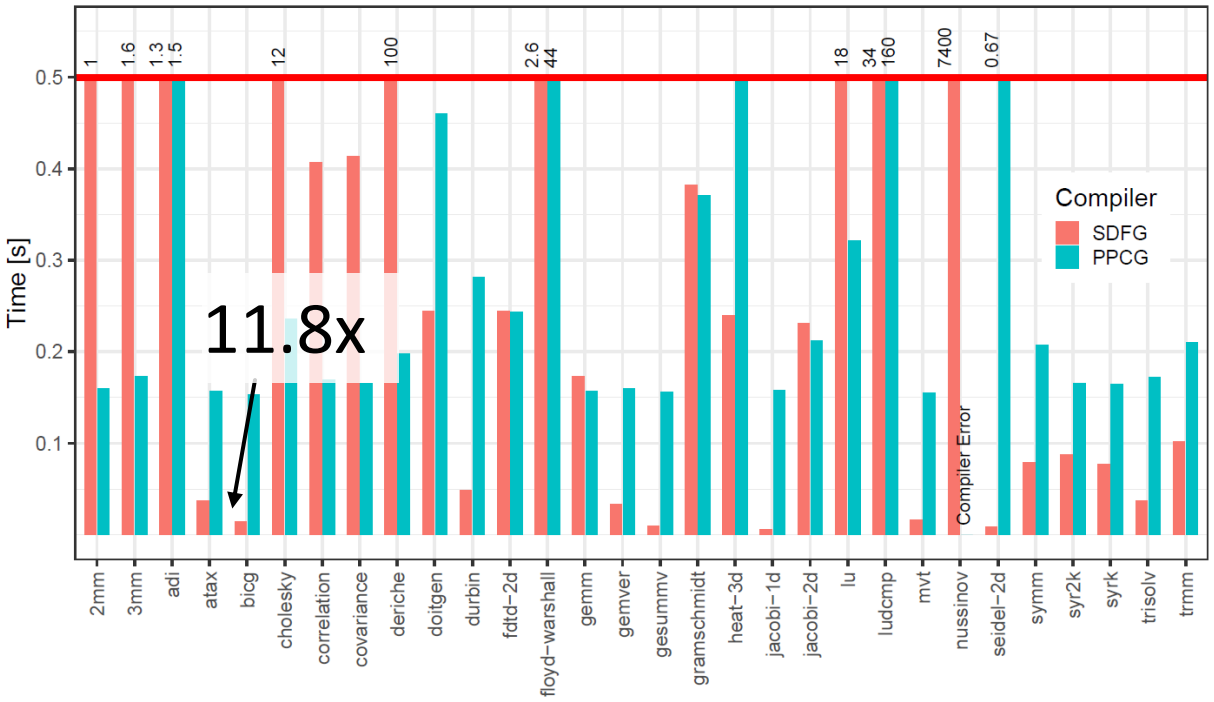
- Polyhedral benchmark with 30 applications

- Without any transformations, achieves 1.43x (geometric mean) over general-purpose compilers



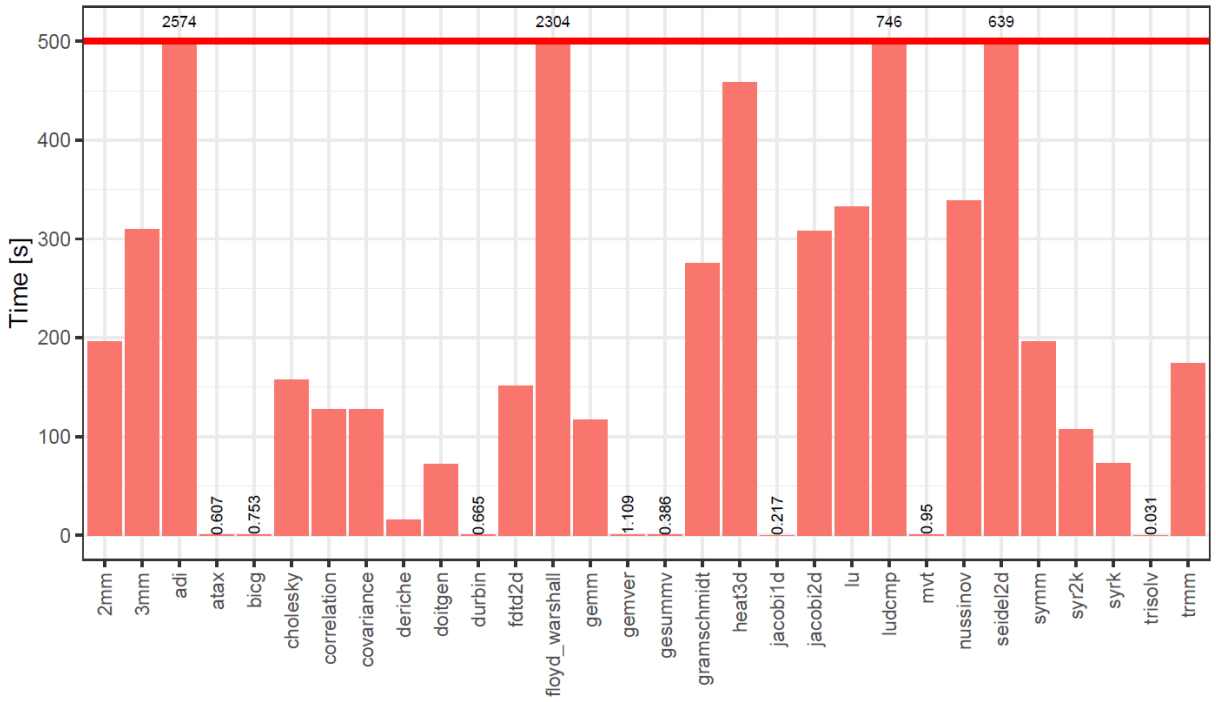
Performance Evaluation: Polybench (GPU, FPGA)

- **Automatically transformed from CPU code**



GPU

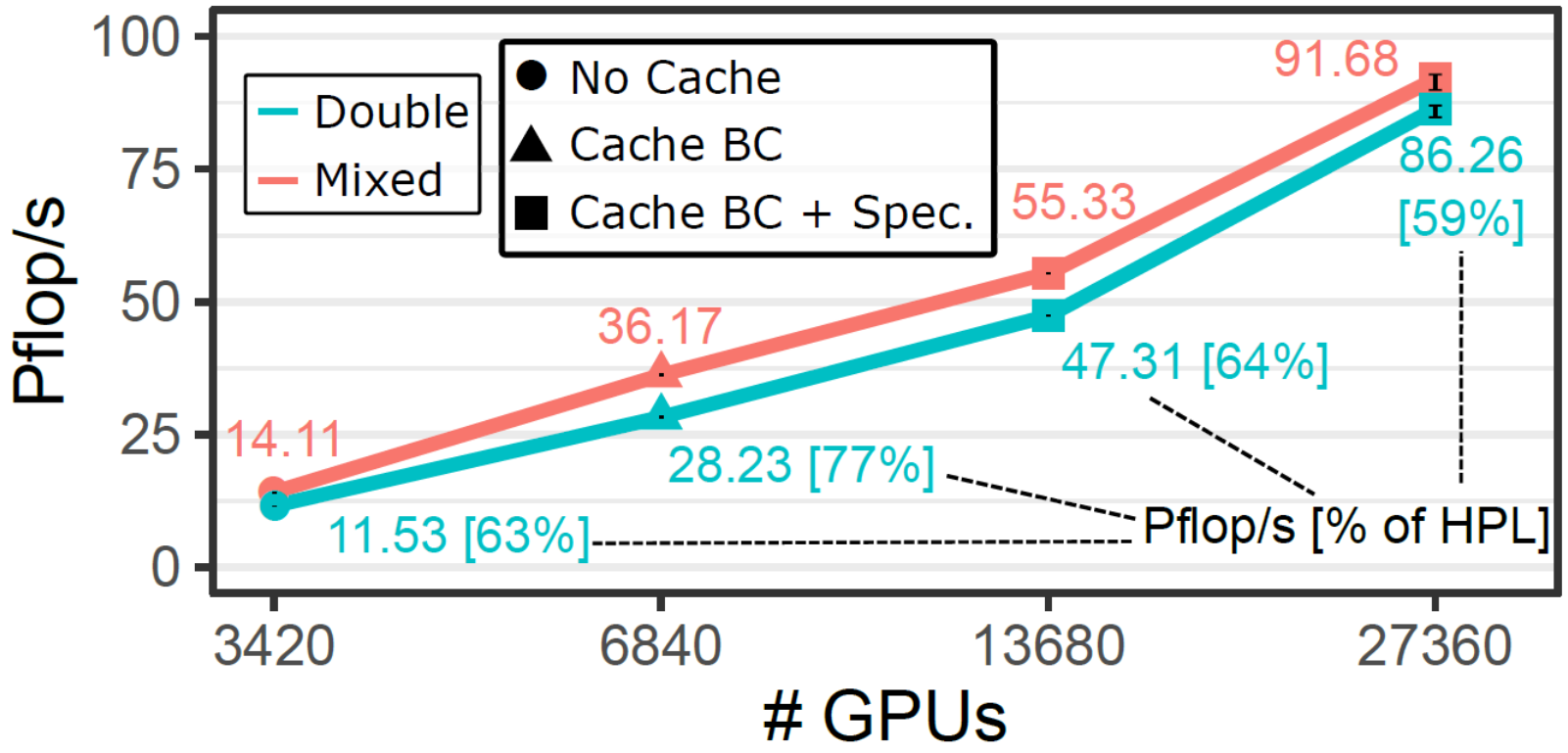
(1.12x geomean speedup)



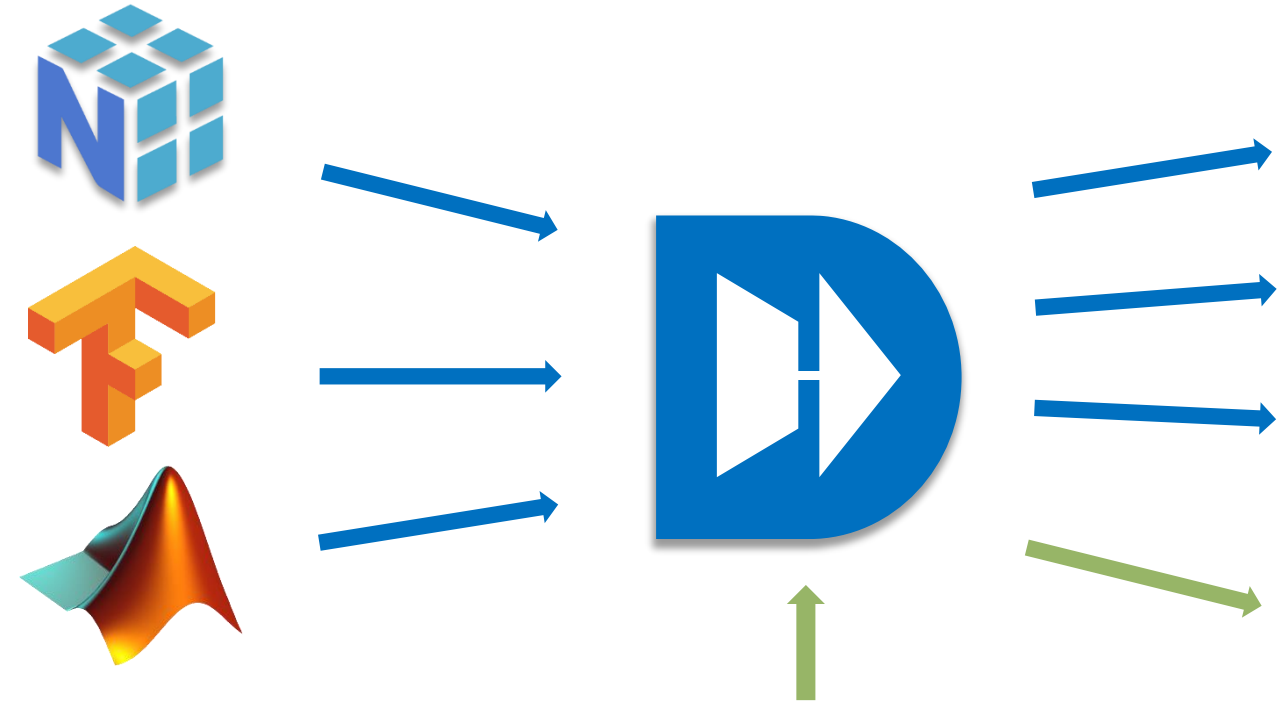
FPGA

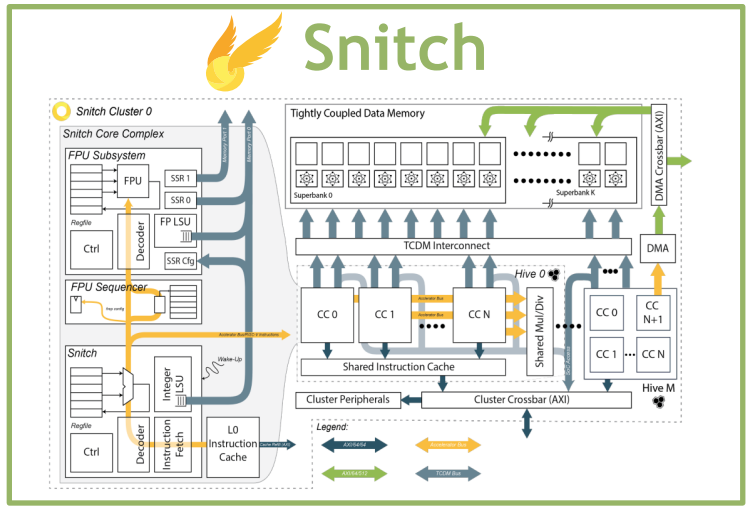
The **first** full set of placed-and-routed Polybench

Performance Evaluation: OMEN

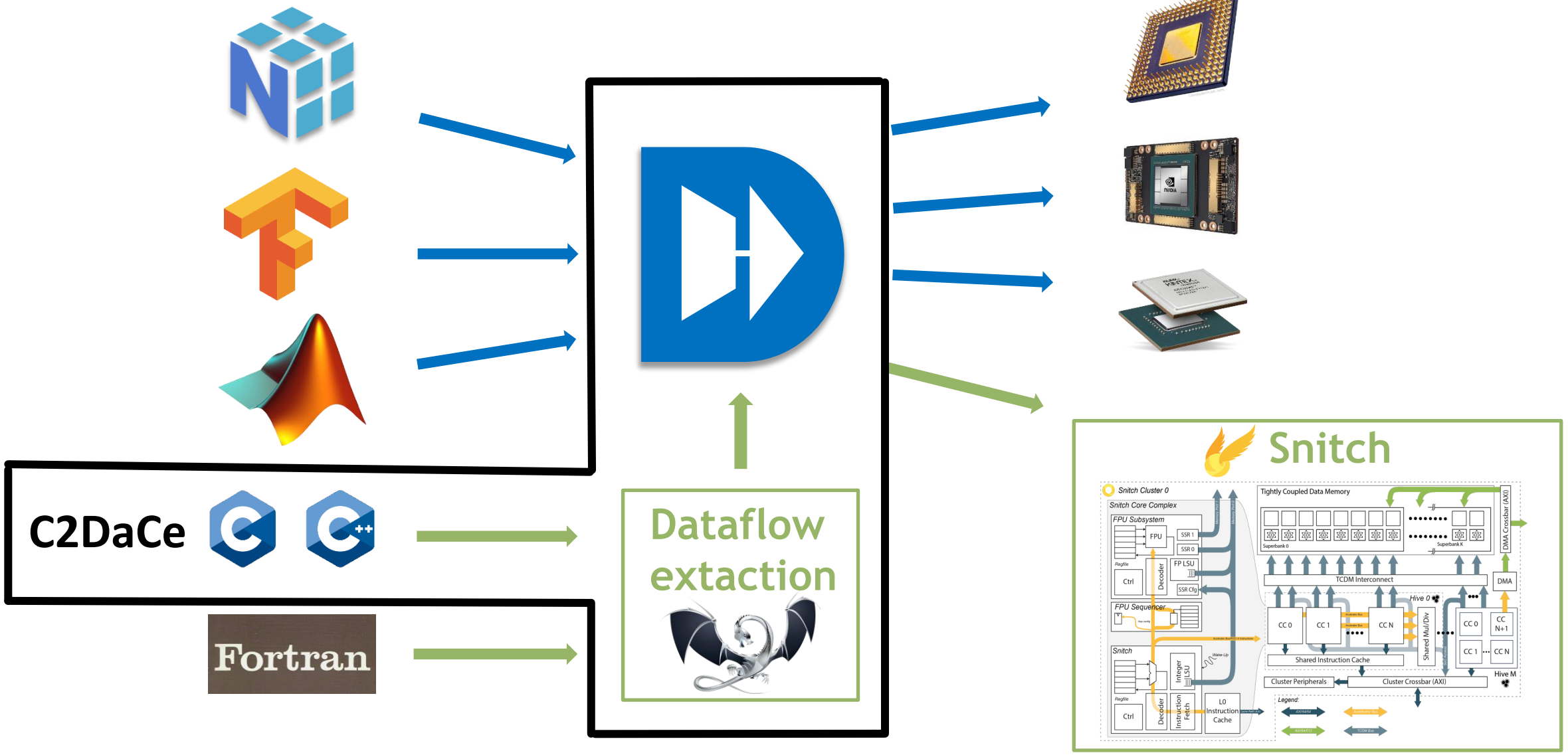


DaFlEx – project overview

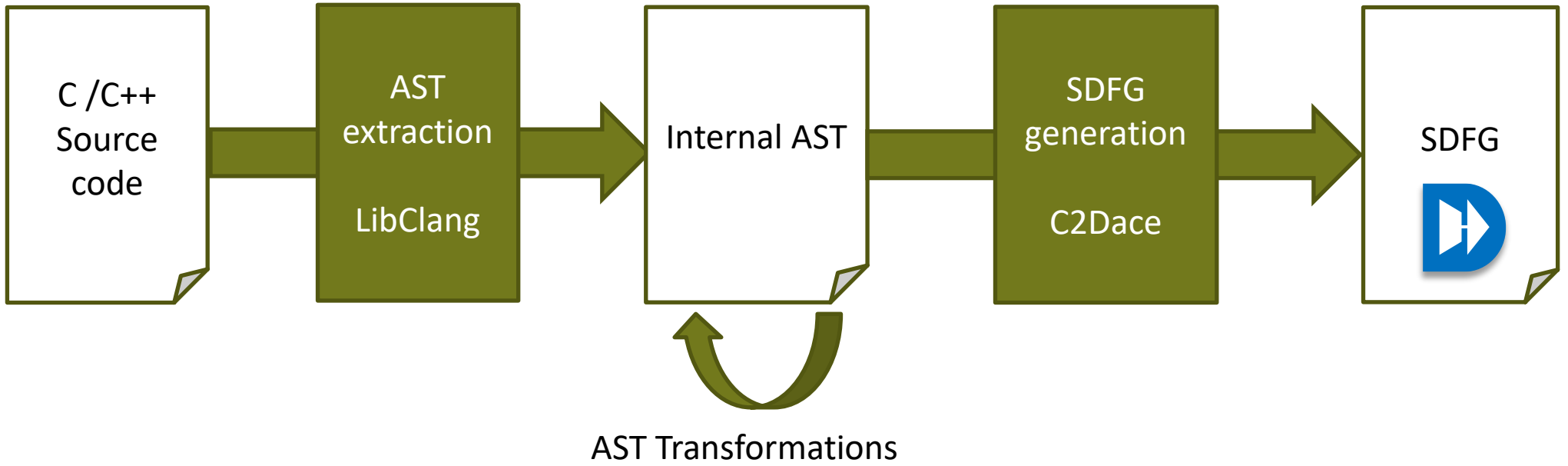




DaFlEx – first goals

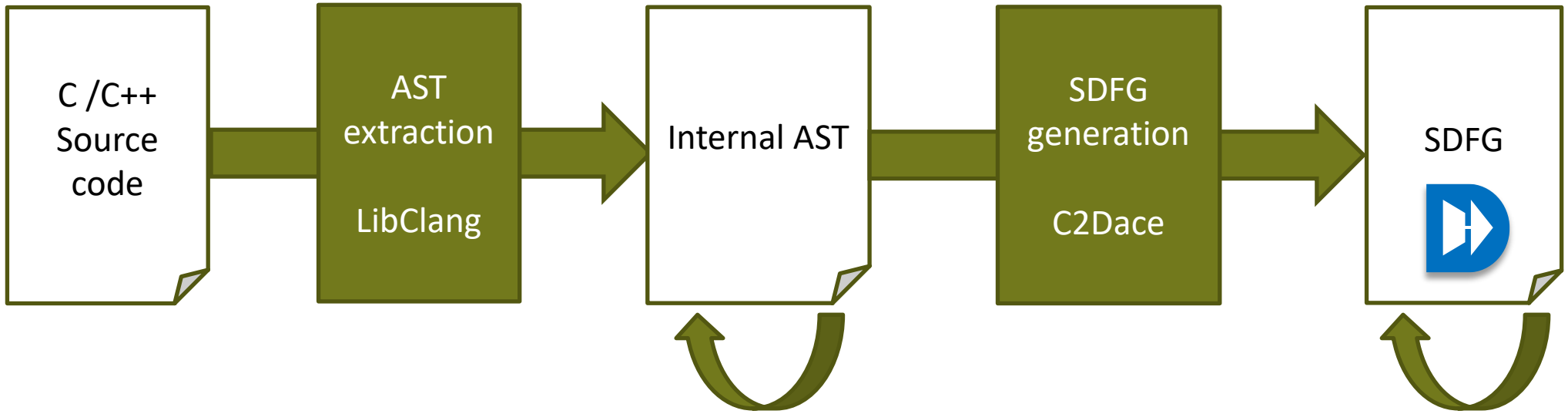


Dataflow extaction pipeline for C/C++



- Array index extraction
- Function call isolation
- Separating variable declaration and initialization

Dataflow extaction pipeline for C/C++



Candidates:

- Polybench
- LULESH
- C HPC codes?

SDFG Transformations

- Symbol promotion
- Create WCR accesses where necessary
- **Loop to Map**

C2DACE

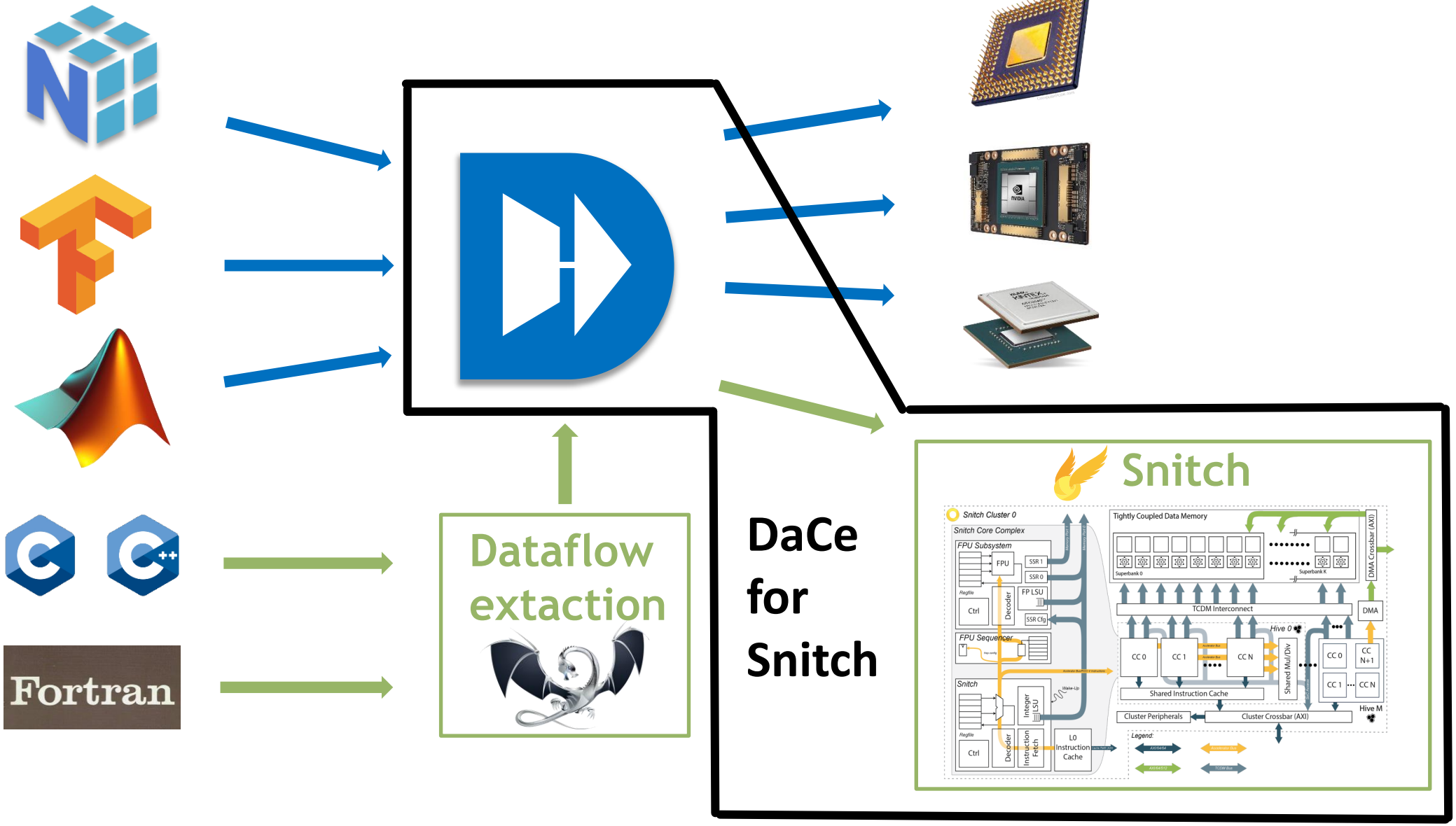
C Language	SDFG Equivalent
Declarations and Types	
Primitive data type	Scalar data container
Type qualifier (const,volatile)	Implicit, hints provided are inherent to the SDFG representation
Storage class (static,auto,register, etc.)	Data container properties (e.g., life-time, storage location)
Pointer	Access node
Array	Array data c
Structure	Data contai
Union	Data contain

C Language	SDFG Equivalent
Expressions and Assignments	
Operators (e.g., Unary,Binary)	Tasklet with incoming and outgoingmemlets for read/written operands
Compound assignments	Tasklet
Array expression	Memlet

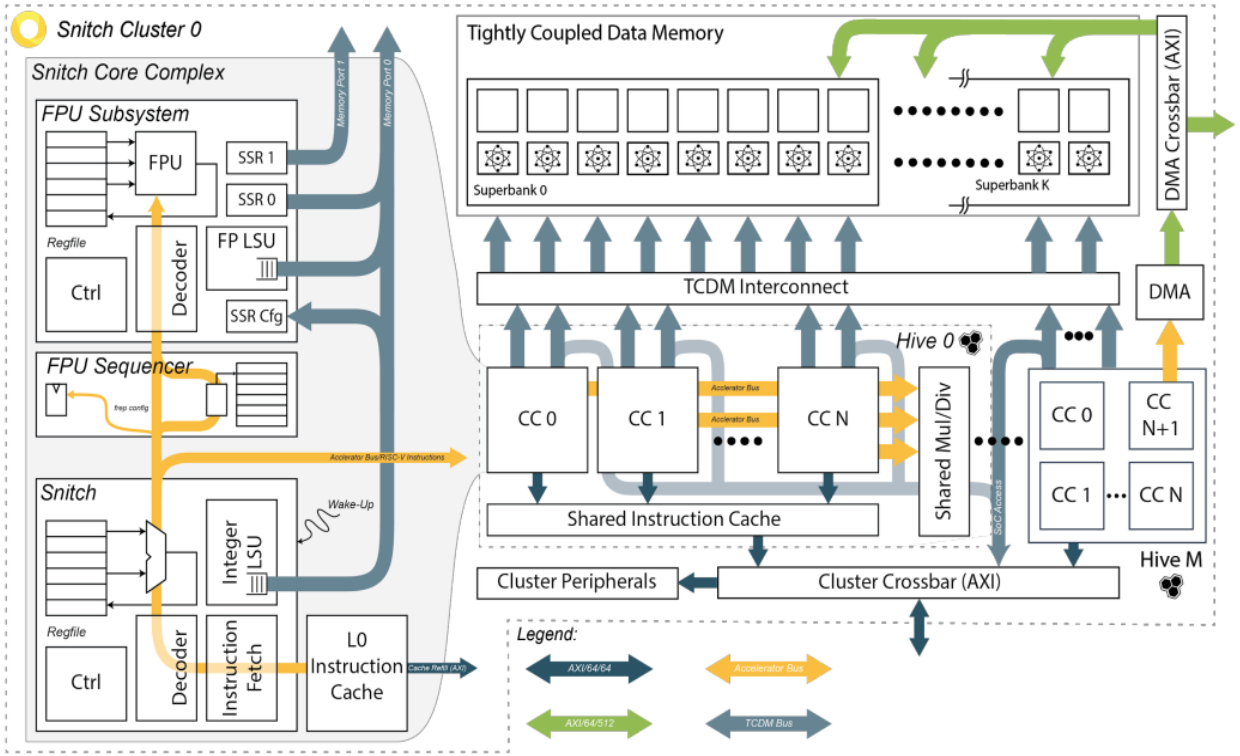
C Language	SDFG Equivalent
Statements	
Compound (blocks)	Nested SDFG
Branching (if,switch)	Branch conditions on state transition edges
Iteration (for, while,do..while)	Nested SDFG for compound state-ment, with states and state transi-tions for loop logic
Function flow	Edge to SDFG exit state, using helpervariable to
	ion if within same SDFG,otherwise d
C Language	alent
Functions	
Function calls (with source)	Nested SDFG for content, memlets reduce shape of inputs and outputs
External/Library calls	Tasklet with library state
Recursion	Unsupported
Function pointers	No equivalent, unsupported

C Language	SDFG Equivalent
Parallelism	
-	**Parametric Map scope**

DaFlEx – first goals



Snitch



RV32IMAFDXfrepXssrXsdma

32 bit RISC-V ISA	Multiplications Divisions	Floating-Point
		Atomics
Custom Extensions		

- **S**treaming **S**emantic **R**egisters
  Register-memory interface
- **DMA** Engine
  Asynchronous Data Movement
- **F**loating Point **Rep**etition
  Hardware loops for the FPU

DaCe code generation for the Snitch

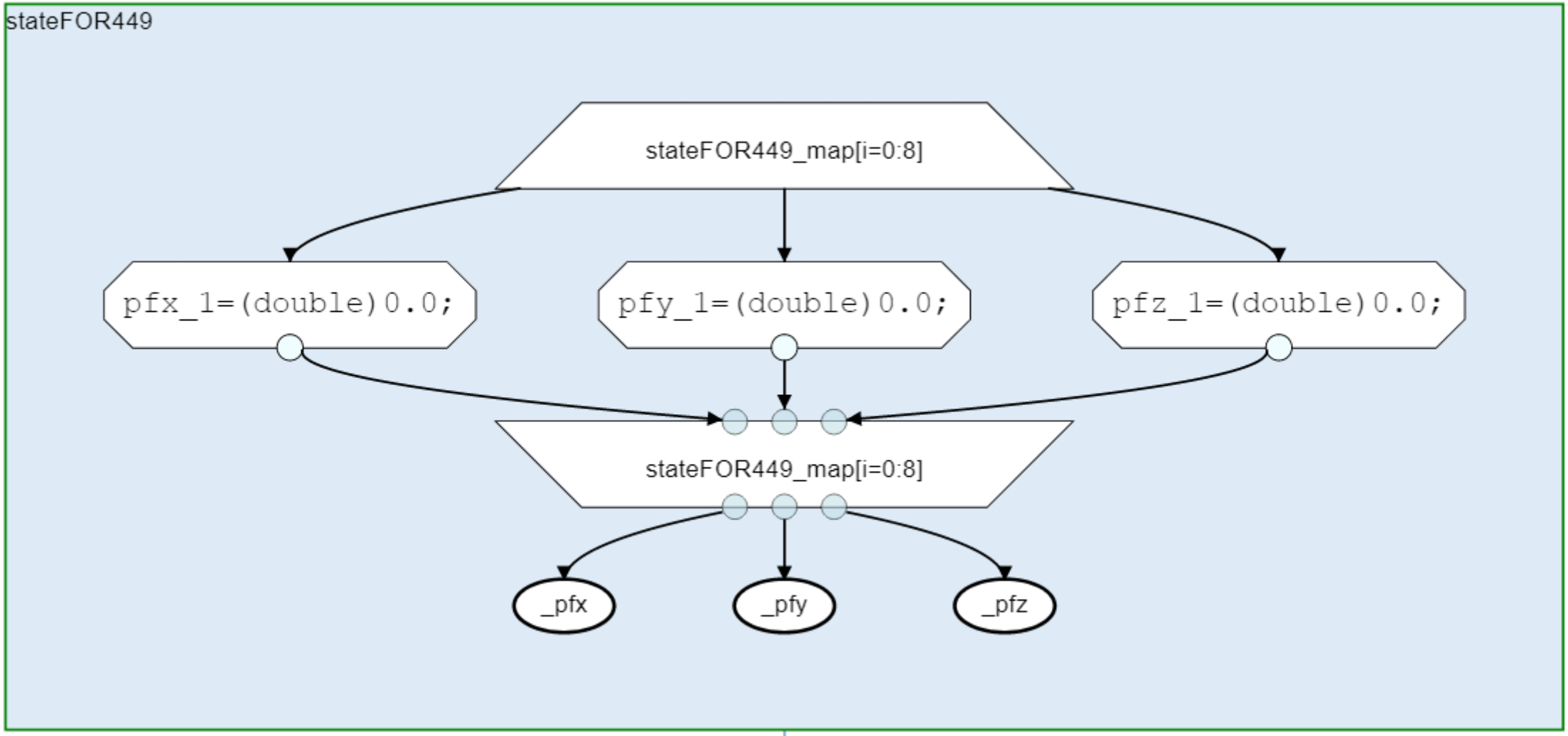
Goal – create a new codegeneration backend to leverage Snitch's custom extensions for data movement, specifically:

TCDM/DMA	Double buffering from host memory to TCDM
SSR	Hot-loop without load/store overhead
Frep	Hardware loops for the FPU

Concept – from an SDFG emit the SSR configuration and express parallelism u
sing OpenMP

```
void __program_daxpy_internal( ... ) {
  __builtin_ssr_setup_1d_r(0, ... );
  __builtin_ssr_setup_1d_r(1, ... );
  __builtin_ssr_enable();
  #pragma omp parallel for
  for (auto i = 0; i < N; i += 1) {
      double in_A = A;
      double in_X = __builtin_ssr_pop(0);
      double in_Y = __builtin_ssr_pop(1);
      double out;
      ////////////////////
      // Tasklet code (multiplication)
      out = ((in_A * in_X) + in_Y);
      ////////////////////
      __builtin_ssr_push(2, out);
  }
  __builtin_ssr_disable();
}
```

Thank you!


stateFOR449
stateFOR449_map[i=0:8]
pfx_1=(double)0.0;
pfy_1=(double)0.0;
pfz_1=(double)0.0;
stateFOR449_map[i=0:8]
_pfx
_pfy
_pfz