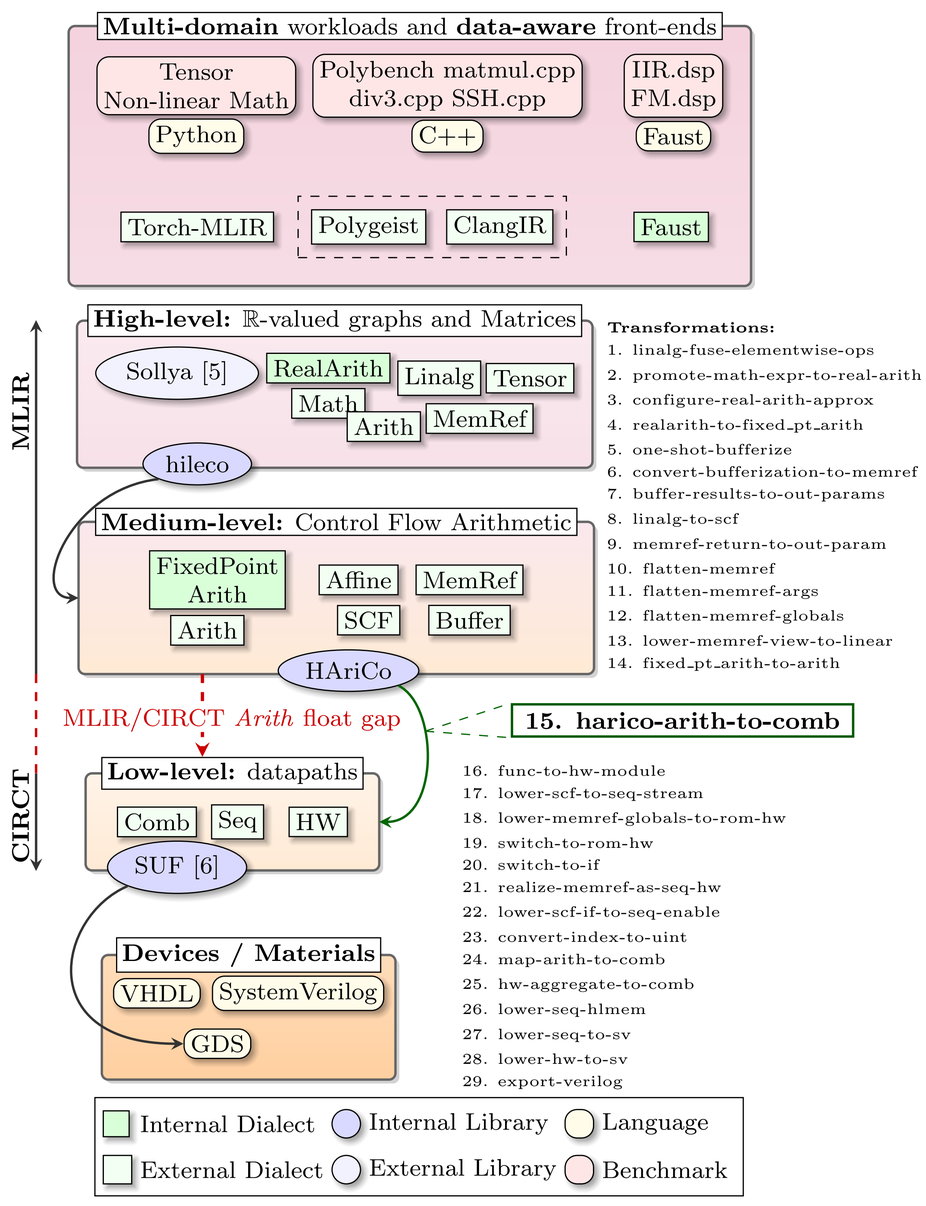
Floating-Point Datapaths in CIRCT via HAriCo and `harico-arith-to-comb` Lowering

Louis Ledoux†, Pierre Cochard†, Florent de Dinechin†
†INSA Lyon, Inria — Emeraude Team

Core contributions

- Bridge MLIR/CIRCT **floating-point arithmetic gap**
- Restructured Core generator ⇒ **HAriCo**
- **Materialization lowering pass**: `harico-arith-to-comb`
- **Multiple lowering strategies**:
  1. Per-op (IEEE754-compliant)
  2. Fused-graph-datapath (e.g. fast-math; fma)
  3. Specialized (Kulisch-style accumulation, squarer)
- **End-to-end flow**: Machine learning, DSP ⇒ **Silicon**

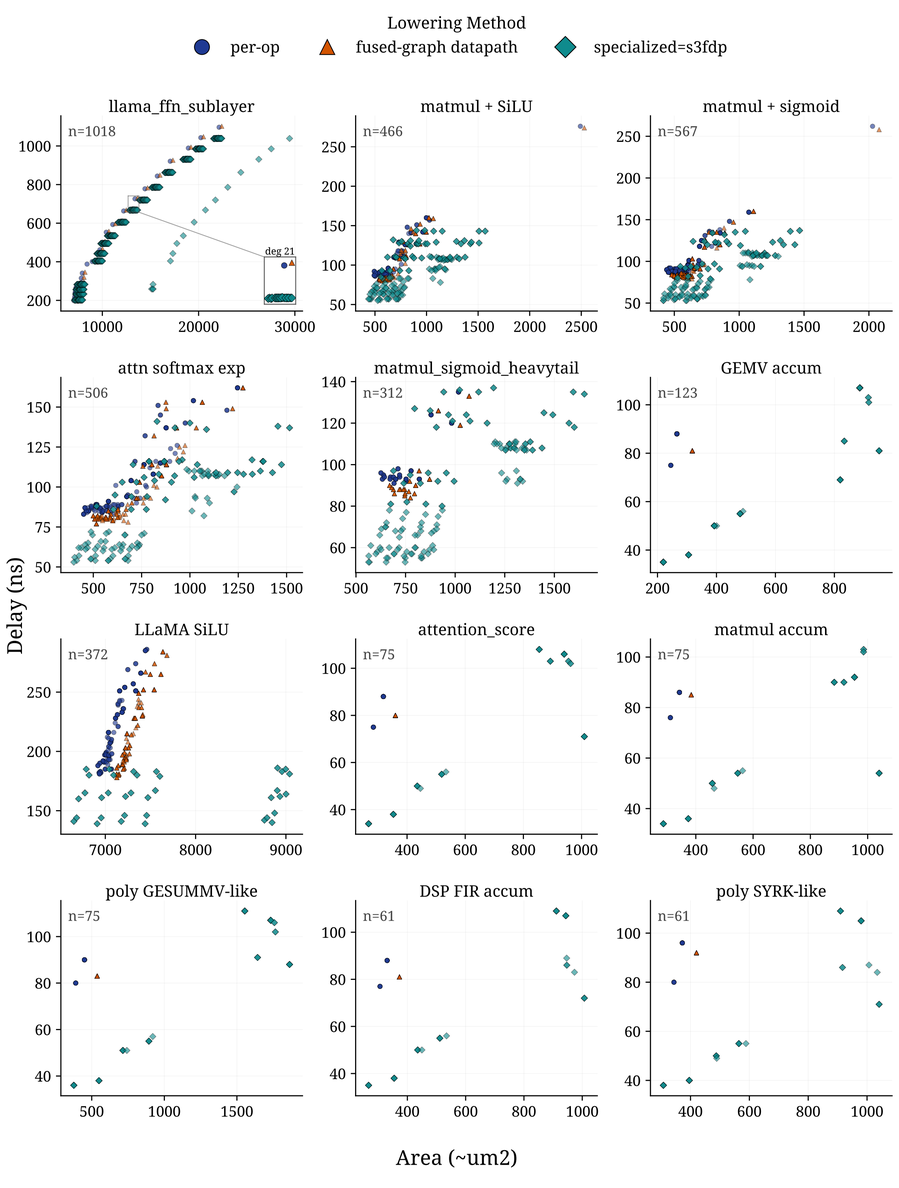
Emeraude-MLIR: Problem to Silicon



Transformations:

1. linalg-fuse-elementwise-ops
2. promote-math-expr-to-real-arith
3. configure-real-arith-approx
4. realarith-to-fixed_pt_arith
5. one-shot-bufferize
6. convert-bufferization-to-memref
7. buffer-results-to-out-params
8. linalg-to-scf
9. memref-return-to-out-param
10. flatten-memref
11. flatten-memref-args
12. flatten-memref-globals
13. lower-memref-view-to-linear
14. fixed_pt_arith-to-arith
15. **harico-arith-to-comb**
16. func-to-hw-module
17. lower-scf-to-seq-stream
18. lower-memref-globals-to-rom-hw
19. switch-to-rom-hw
20. switch-to-if
21. realize-memref-as-seq-hw
22. lower-scf-if-to-seq-enable
23. convert-index-to-uint
24. map-arith-to-comb
25. hw-aggregate-to-comb
26. lower-seq-hlmem
27. lower-seq-to-sv
28. lower-hw-to-sv
29. export-verilog

Results from DSE



From MLIR to CIRCT: `harico-arith-to-comb`

1 Per-op

- IEEE754-compliant
- 1 Operation → 1 Operator
- N Operations → N Roundings

2 Fused-graph-datapath

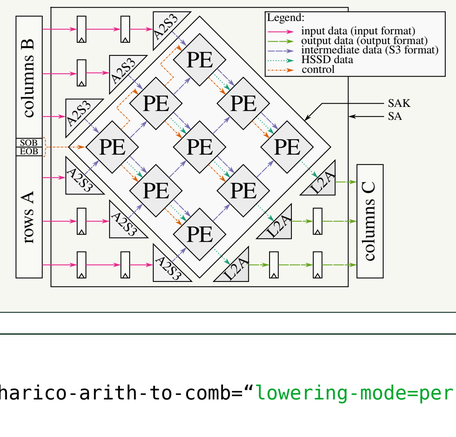
- FMA philosophy
- 1 deferred rounding
- Expression fusion

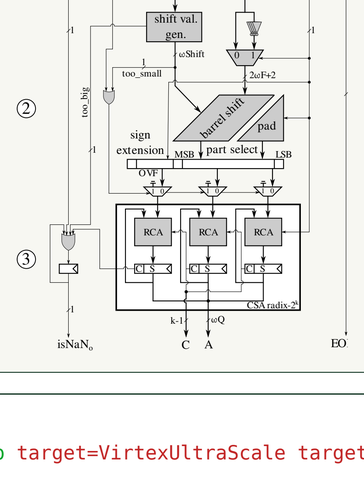
$$\frac{1}{\sqrt{x^2+y^2}}$$

$$\sin(\omega t + \varphi)$$

3 Specialized

- Squarers [1]
- Constant multipliers [2]
- Kulisch / Systolic Arrays [3]
- Target semantics





Transformations

1. `emeraude-mlir-opt fpmult-loop-muladd.mir --harico-arith-to-comb="lowering-mode=per-op target=VirtexUltraScale target-frequency=2.5e7"`
2. `emeraude-mlir-opt fpmult-loop-muladd.mir --harico-arith-to-comb="lowering-mode=fused-graph-datapath target=sky130 target-frequency=2.5e7"`
3. `emeraude-mlir-opt fpmult-loop-muladd.mir --harico-arith-to-comb="lowering-mode=specialized specializations=enable=s3fdp,s3fdp.chunk_size=32,s3fdp.ovf=10,s3fdp.msb=10,s3fdp.lsb=-20 target=tsmc7 target-frequency=2.5e7"`

E upstream IR (scf/arith)

```
module {
  func.func @fpmult_loop_muladd_s3fdp(
    %clk: i1 {hw.name = "clk"},
    %reset: i1 {hw.name = "reset"},
    %a: memref<2x2xf32> {hw.name = "a"},
    %b: memref<2x2xf32> {hw.name = "b"},
    %c: memref<1xf32> {hw.name = "c"})
  -> (f32 {hw.name = "r"}) {
    %c0 = arith.constant 0 : index
    %c1 = arith.constant 1 : index
    %c2 = arith.constant 2 : index
    scf.for %i = %c0 to %c2 step %c1 {
      scf.for %j = %c0 to %c2 step %c1 {
        %x = memref.load %a[%i, %j] : memref<2x2xf32>
        %y = memref.load %b[%i, %j] : memref<2x2xf32>
        %acc = memref.load %c[%c0] : memref<1xf32>
        %m = arith.mulf %x, %y : f32
        %s = arith.addf %acc, %m : f32
        memref.store %s, %c[%c0] : memref<1xf32>
      }
    }
    %r = memref.load %c[%c0] : memref<1xf32>
    func.return %r : f32
  }
}
```

C CIRCT (comb/hw/seq)

```
[...]
%c0_i23 = hw.constant 0 : i23
%c0_i8 = hw.constant 0 : i8
%c0 = arith.constant 0 : index
%c1 = arith.constant 1 : index
%c2 = arith.constant 2 : index
  scf.for %arg5 = %c0 to %c2 step %c1 {
    scf.for %arg6 = %c0 to %c2 step %c1 {
      %1 = memref.load %arg2[%arg5, %arg6] : memref<2x2xi32>
      %2 = memref.load %arg3[%arg5, %arg6] : memref<2x2xi32>
      %3 = memref.load %arg4[%c0] : memref<1xi32>
      %4 = comb.extract %1 from 23 : (i32) -> i8
      %5 = comb.extract %1 from 0 : (i32) -> i23
      %6 = comb.icmp eq %4, %c0_i8 : i8
[...]
```

F Func. specs.

- domains
- operator
- i/o format
- $f(x) = \sin(x)$

P Perf. specs.

- frequency
- asic pdk
- fpga model
- power

Specifications

Arithmetic Patterns

I HAriCo

1 Inputs
- MLIR
- Pattern matching

2 IR/Passes
- Scheduling
- Pipelining
- Validation

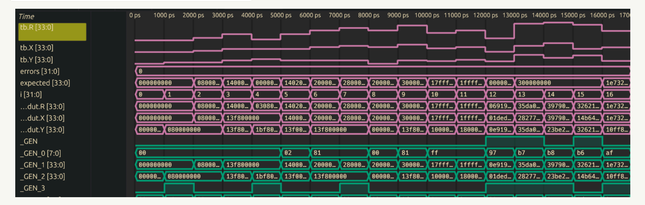
PassManager

3 Generation
- Conversion
- Testbenches
- Verilog
- VHDL

Emit CIRCT

GNU MPFR [4]

W (Surfer) Waveform



Tapeouts

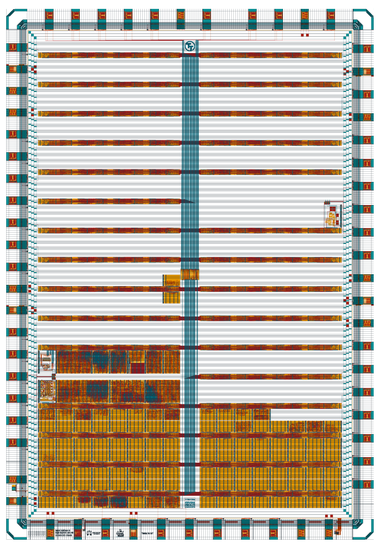


Figure 2: MLIR S3FDP TTSKY26A/130

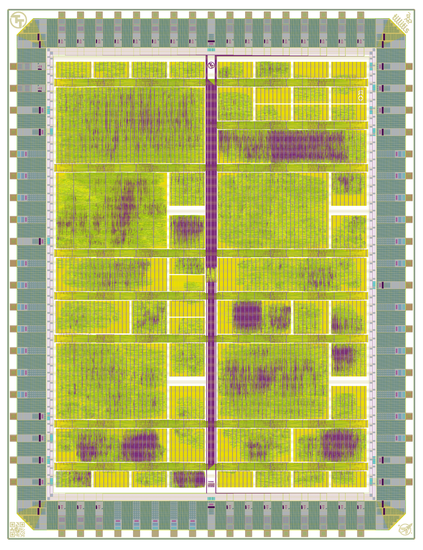


Figure 3: Faust Softclipping TTGF0p2/180

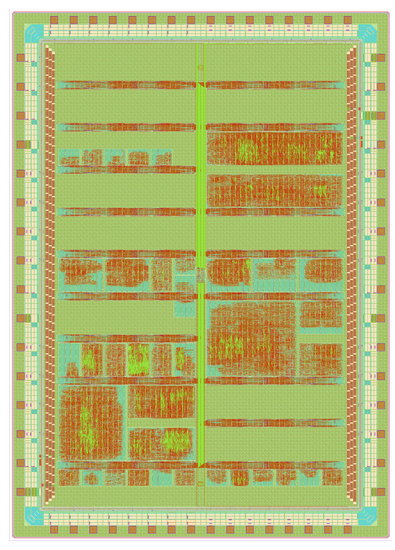


Figure 4: MLIR per-op+S3FDP TTIHP0p4/130

Bibliography & Sources

Tapeout repository

[1] F. de Dinechin and M. Kumm, *Application-Specific Arithmetic*. Springer.

[2] B. Barbe, L. Ledoux, A. Volkova, and F. de Dinechin, "Reconfigurable constant multipliers: Hardware models, optimization algorithm and applications," *Microprocessors and Microsystems*, p. 105270, 2026, doi: https://doi.org/10.1016/j.micpro.2026.105270.

[3] L. Ledoux and M. Casas, "An Open-Source Framework for Efficient Numerically-Tailored Computations," in *2023 33rd International Conference on Field-Programmable Logic and Applications (FPL)*, 2023, pp. 19–26. doi: 10.1109/FPL60245.2023.00011.

[4] L. Fousse, G. Hanrot, V. Lefèvre, P. Pélissier, and P. Zimmermann, "MPFR: A Multiple-Precision Binary Floating-Point Library with Correct Rounding.," *ACM Transactions on Mathematical Software*, vol. 33, no. 2, p. article13, 2007, doi: 10.1145/1236463.1236468.

[5] S. Chevillard, M. Joldeş, and C. Lauter, "Sollya: an environment for the development of numerical codes," in *Proceedings of the Third International Congress Conference on Mathematical Software*, in ICMS'10. Kobe, Japan: Springer-Verlag, 2010, pp. 28–31.

[6] L. Ledoux and M. Casas, "The Grafted Superset Approach: Bridging Python to Silicon with Asynchronous Compilation and Beyond," in *2024 4th Workshop on Open-Source Design Automation (OSDA), hosted at the International Conference on Design, Automation and Test in Europe Conference (DATE),*