Digital mixed-language simulators

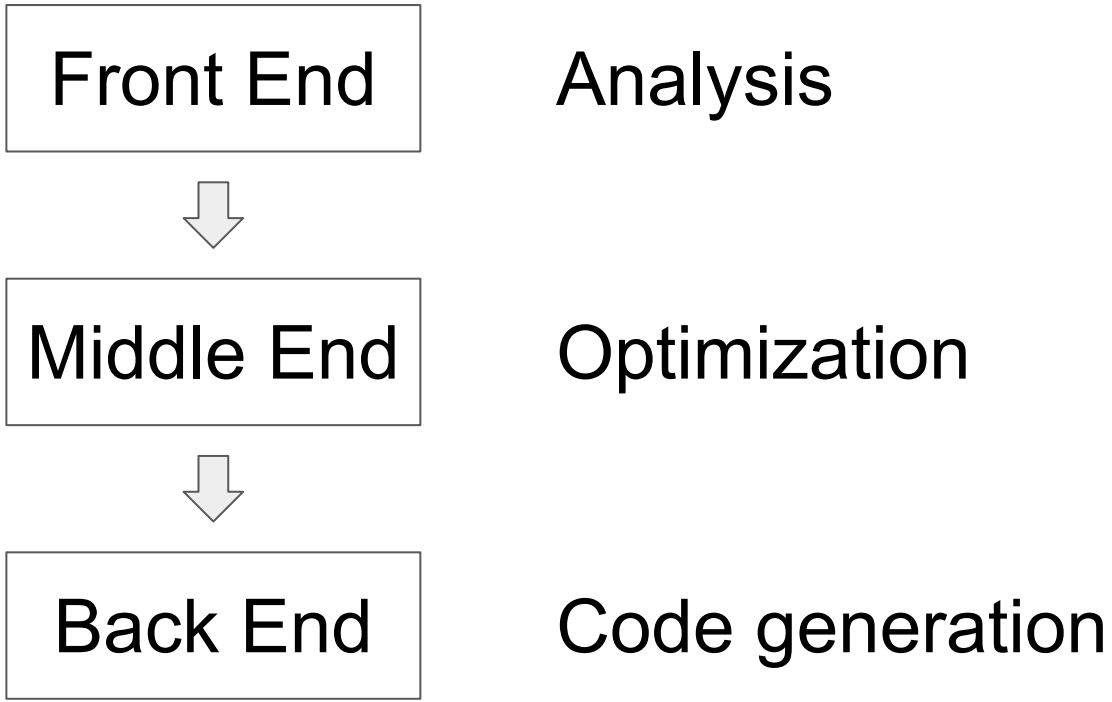
Architectures and implementations

Michele Castellana
CERN
michele.castellana@cern.ch
FOSDEM 2016

Digital mixed-language simulator

- What is a simulator?
- Event based
- Why a mixed language simulator?

Compiler: Overview Architecture



Intermediate Representation

- Aid the analysis
- Different levels of abstraction
    - Example: AST and RTL
- Structure depends on the abstraction

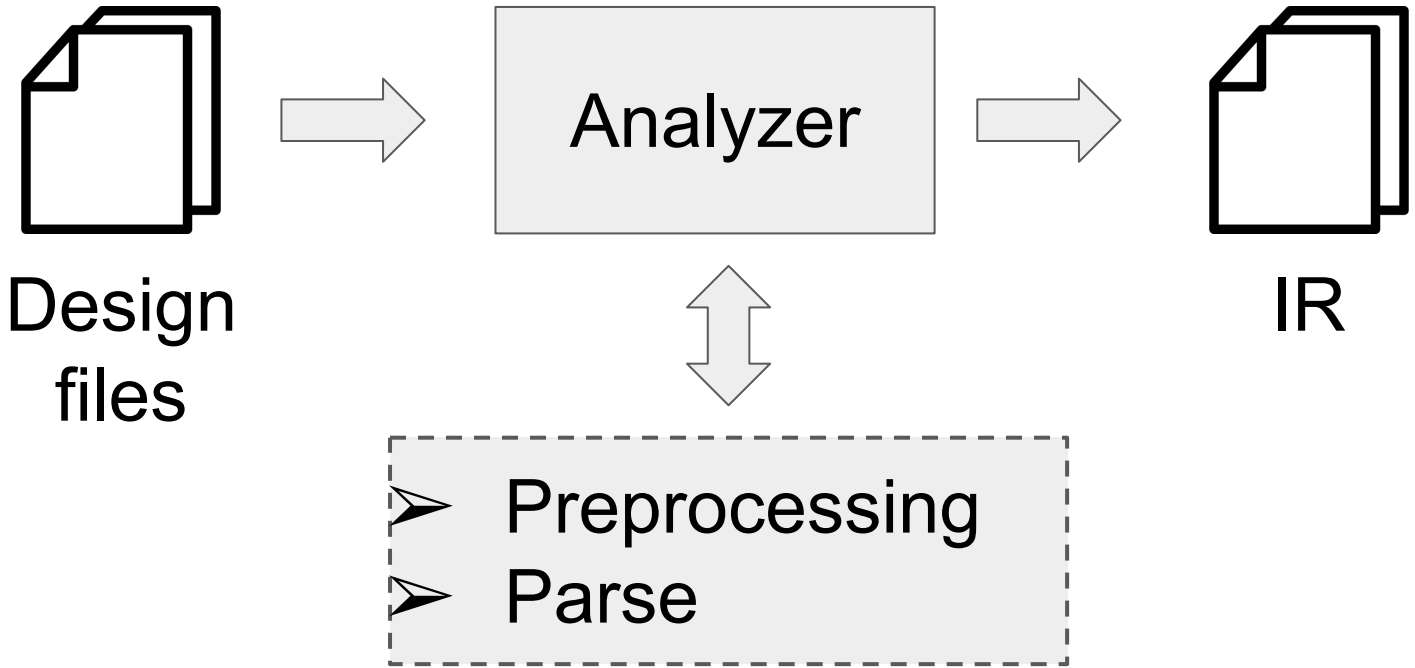
Original code	High-level IR	Mid-level IR	Low-level IR
int a[10][20]; a[i][j+2];	t1 = a[i, j+2]	t1 = j + 2 t2 = i * 20 t3 = t1 + t2 t4 = 4 * t3 t5 = addr a t6 = t5 + t4 t7 = *t6	r1 = [fp - 4] r2 = [r1 + 2] r3 = [fp - 8] r4 = r3 * 20 r5 = r4 + r2 r6 = 4 * r5 r7 = fp – 216 f1 = [r7 + r6]

HDL compiler

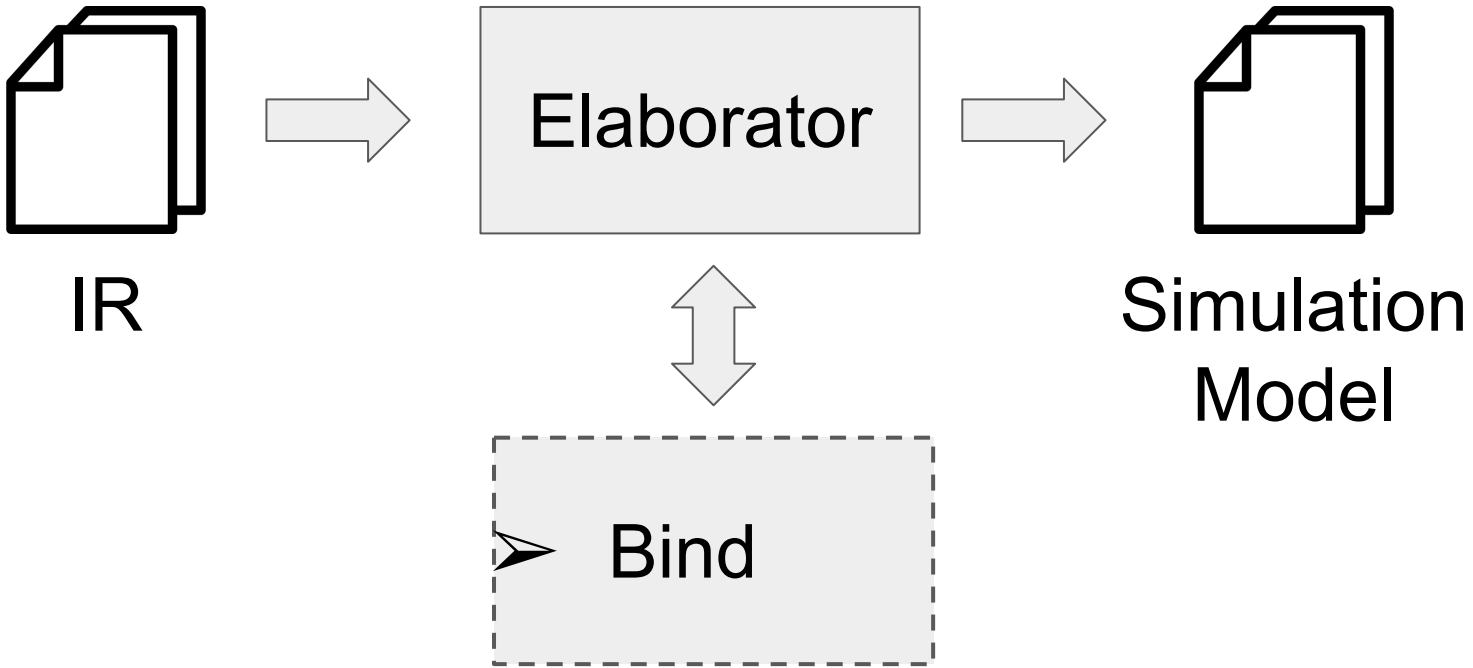
Three logical parts:

- Analysis
- Elaboration
- Simulation

Analysis



Elaboration



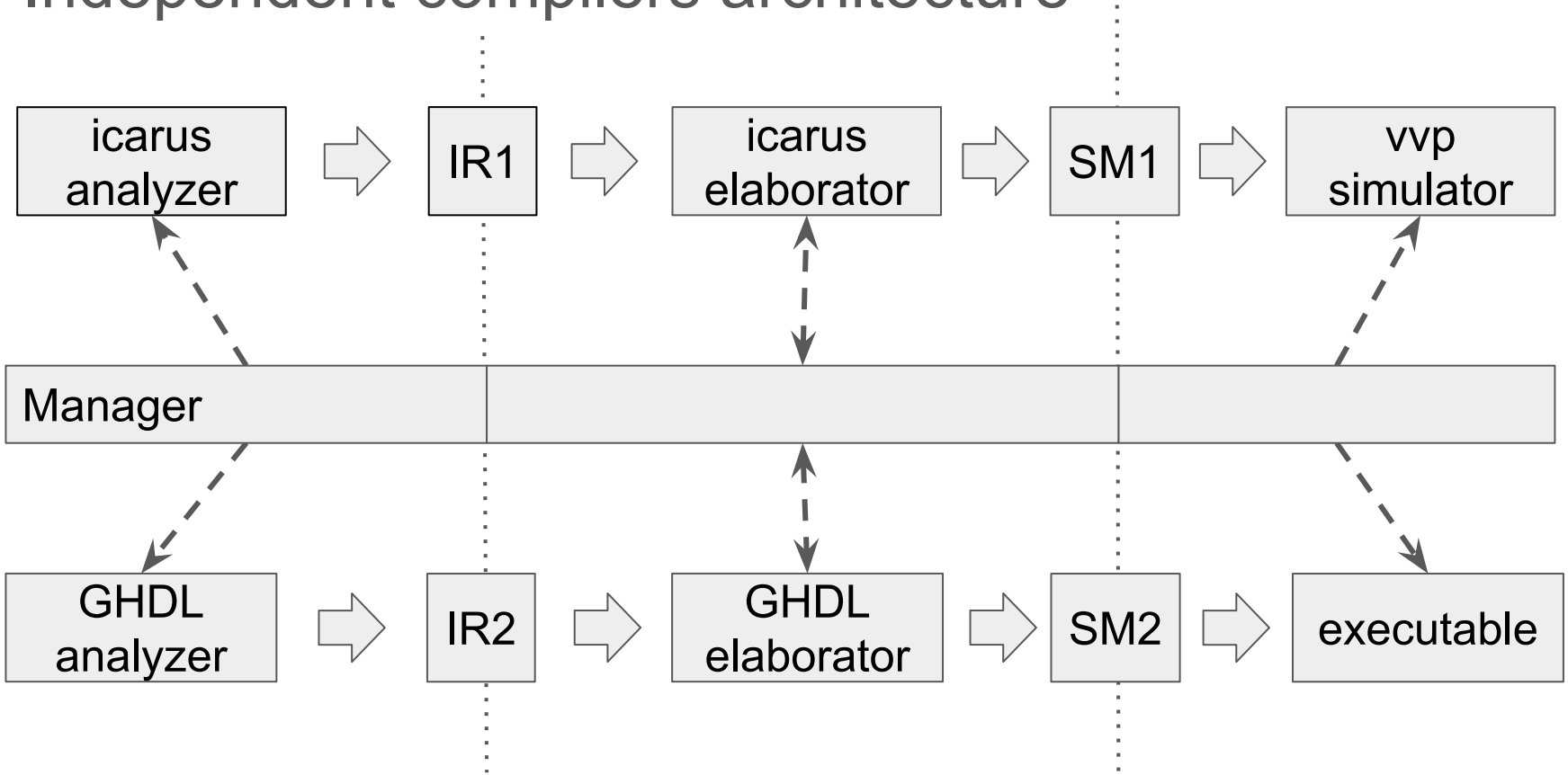
Simulation

- Different possibilities
  - Runtime library
  - Virtual Machine
  - Produce code compliant with a simulation library

Digital Mixed-language simulator

Two main proposed ideas

Independent compilers architecture



Pros & cons

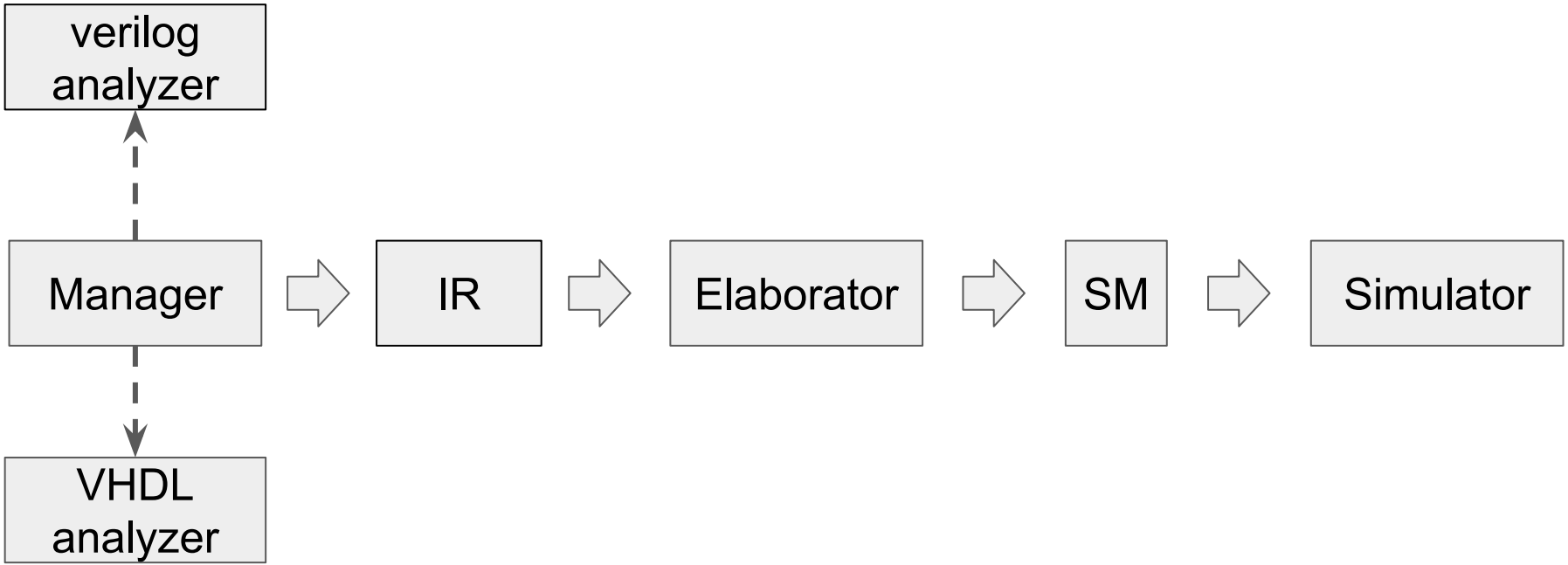
Pros

- Code reusability
- Short-term results

Cons

- Different IRs and elaborator
- Long-term features deal with synchronization
- Maintainability
- Worst case scenario

Savant architecture



Pros & cons

Pros

- Long-term results
  - No worst case scenario
  - Unique IR and elaborator
  - Maintainability

Cons

- Code reusability
- Short-term results

Thank you!

Questions?

References

- GHDL
- Icarus Verilog
- Savant project