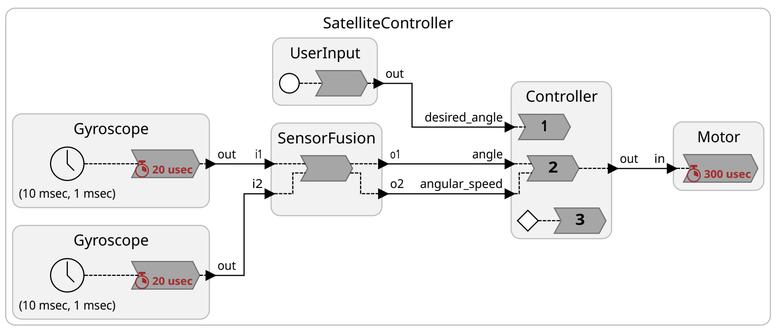
Dresden University of Technology

CHAIR FOR COMPILER CONSTRUCTION

Compiling the Reactor Model of Computation for Many-Core Systems

Contact: Tassilo Tanneberger, tassilo.tanneberger@tu-dresden.de

Lingua Franca and the Reactor Model of Computation




```
1  reactor SatelliteController {                                    uLF
2    gyro_1 = new Gyroscope()
3    gyro_2 = new Gyroscope()
4    sensor_fusion = new SensorFusion()
5    user_input = new UserInput()
6    controller = new Controller()
7    motor = new Motor()
8    // Connecting the components
9    gyro_1.out -> sensor_fusion.i1
10   gyro_2.out -> sensor_fusion.i2
11   sensor_fusion.o1 -> controller.angle
12   sensor_fusion.o2 -> controller.angular_speed
13   user_input.out -> controller.desired_angle
14   controller.out -> motor.in
15 }
```

```
1  reactor Gyroscope {                                              uLF
2    timer t(10msec, 1msec);
3    output out: float
4
5    reaction(t) -> out {=
6      float sensor_reading = i2c_read_reg(...);
7      lf_set(out, sensor_reading);
8    =} deadline(20msec) {=
9      // Deadline miss handler: applies a delayed correction to the sensor reading
10   =}
11 }
```

Determinism

The reactor semantics guarantee that, given the same inputs and external timing, a program always produces the same outputs—even when running on multiple cores or distributed machines.

Concurrency and Parallelism

Dependencies between reactions are declared explicitly, so independent reactions can execute in parallel on multiple cores or across distributed nodes without compromising determinism.

Declarative

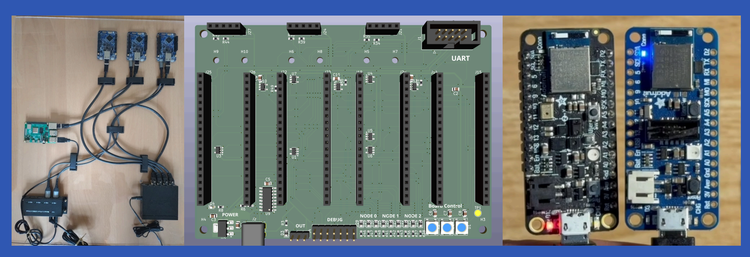
LF allows declaring reactor structure, ports, connections, and time semantics - the behavior, not how to schedule it. The compiler then synthesizes target-language code, including the networking code.

micro-LF: Runtime for Networked Embedded Systems

- Runtime for executing micro-LF applications on resource-constrained microcontrollers.
- Supports distributed execution via various communication protocols, e.g., TCP/IP, UART, BLE.
- Low memory footprint and minimal binary overhead.

https://github.com/lf-lang/reactor-uc
https://micro-lf.org

Testbeds for the Framework



TCP/IP **UART** **BLE**

Testing the framework across various heterogeneous scenarios.

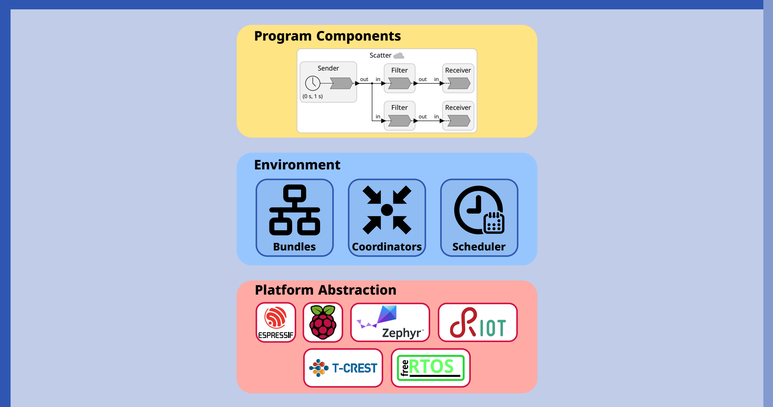


Figure 1: micro-LF Runtime Structure

RFlow: MLIR-based IR for Compile-Time Optimization and Static Scheduling

- Implements compile-time optimizations such as dead-code elimination, constant propagation, and connection compression.
- Typically, a runtime implements the execution semantics, but a static schedule can also be generated at compile time.
- Requires compilation heuristics for optimal core mapping and DAG scheduling.

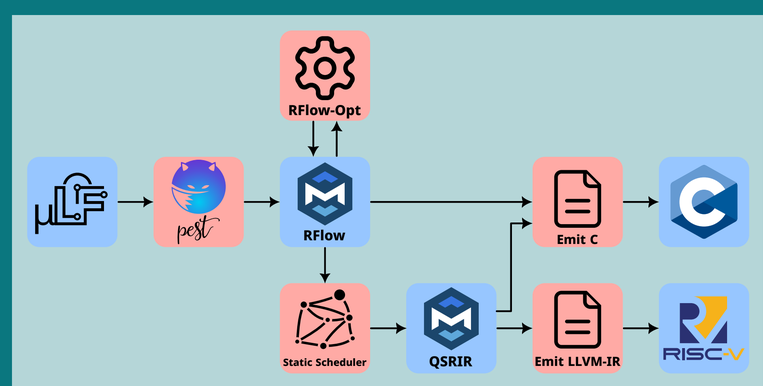


Figure 2: MLIR-based Compilation Flow

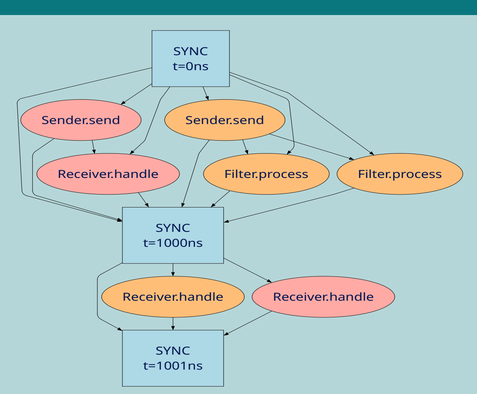


Figure 3: Example DAG for Static Scheduling

RISC-V Multi-Core Architecture for Efficient Execution of the Reactor MoC

micro-LF on the Patmos Processor

- Patmos is a time-predictable multi-core processor.
- The platform uses S4NoC, which provides a fixed schedule for transmitting data between cores.
- micro-LF supports building programs that use this NoC.

This work about Patmos is led by Ehsan Khodad and Martin Schoeberl from DTU

PRET Instructions for RISC-V

Mnemonic	Description
`DU op1, op2`	Delay until the physical clock reaches the sum of a base timepoint (op1) and an offset (op2).
`WLT op1, op2`	Wait until the value of a variable (op1) becomes less than a threshold (op2).
`WU op1, op2`	Wait until the value of a variable (op1) is greater than or equal to a threshold (op2).

Intra-Core Synchronization and Atomics

- Which RISC-V extensions provide similar timing and synchronization functionality?
- How can atomics and inter-core synchronization be best implemented?
- Which RISC-V core provides many of prerequisites to build upon?

In Collaboration With or Funded By:

Berkeley

cfaed CENTER FOR ADVANCING ELECTRONICS DRESDEN

DRESDEN concept

WR WISSENSCHAFTSRAT

DFG Deutsche Forschungsgemeinschaft