Open-Source Modelling and Simulation of Innovative Power Generation Systems Using OpenModelica: The Case of the FlexiCaL Project

Francesco Casella

Politecnico di Milano, Italy

OSMC Vice-Director

(francesco.casella@polimi.it)

POLITECNICO MILANO 1863

OpenModelica

Outline

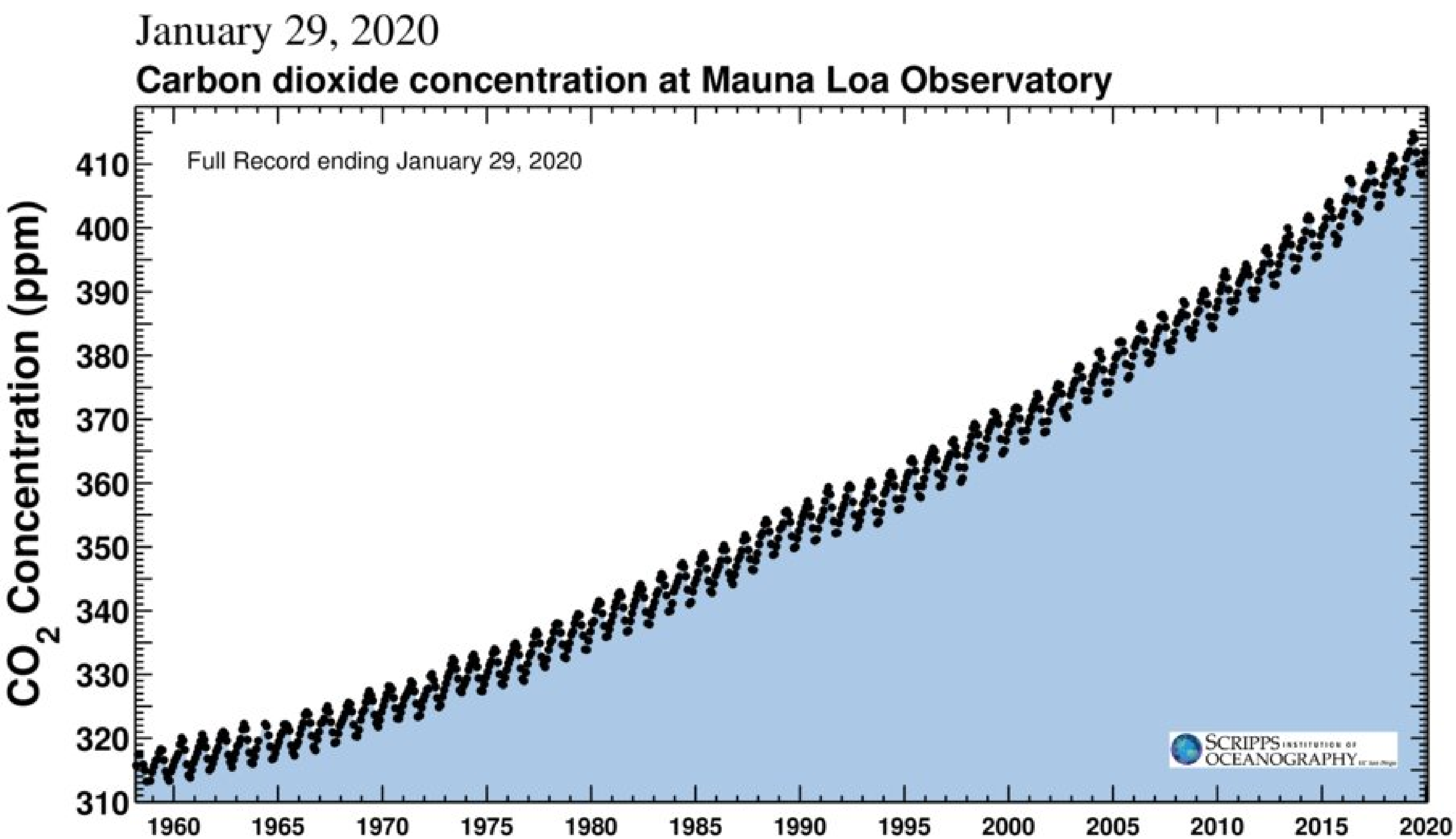
- The issue of climate change
- The FlexiCaL project

- Objectives of the modelling activity
- What are the Modelica models used for
- Relevance for the OSMC and the scientific community
- Presentation of the model

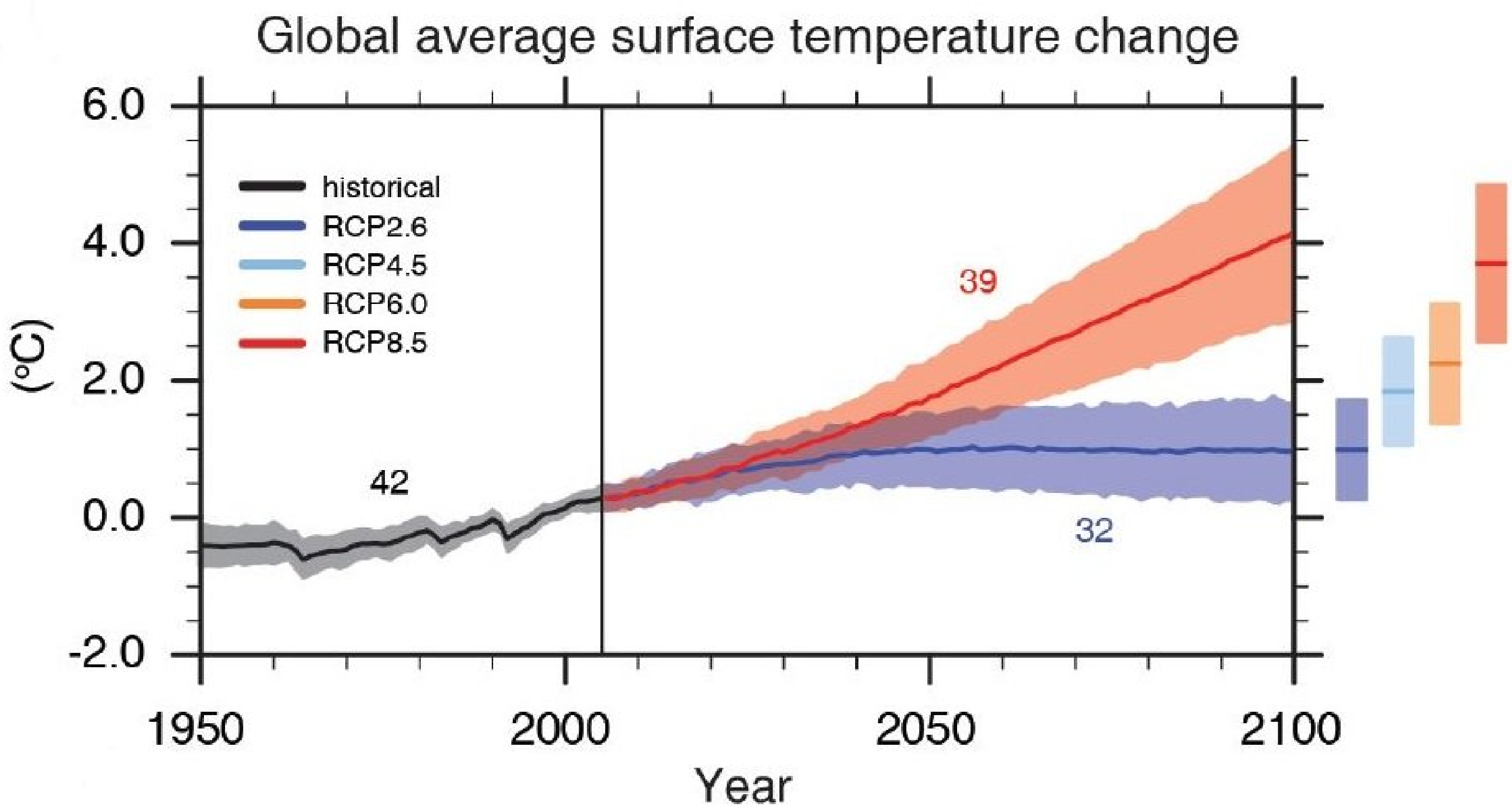
- Current status of OMC support
- Outlook and future work

The Issue of Climate Change

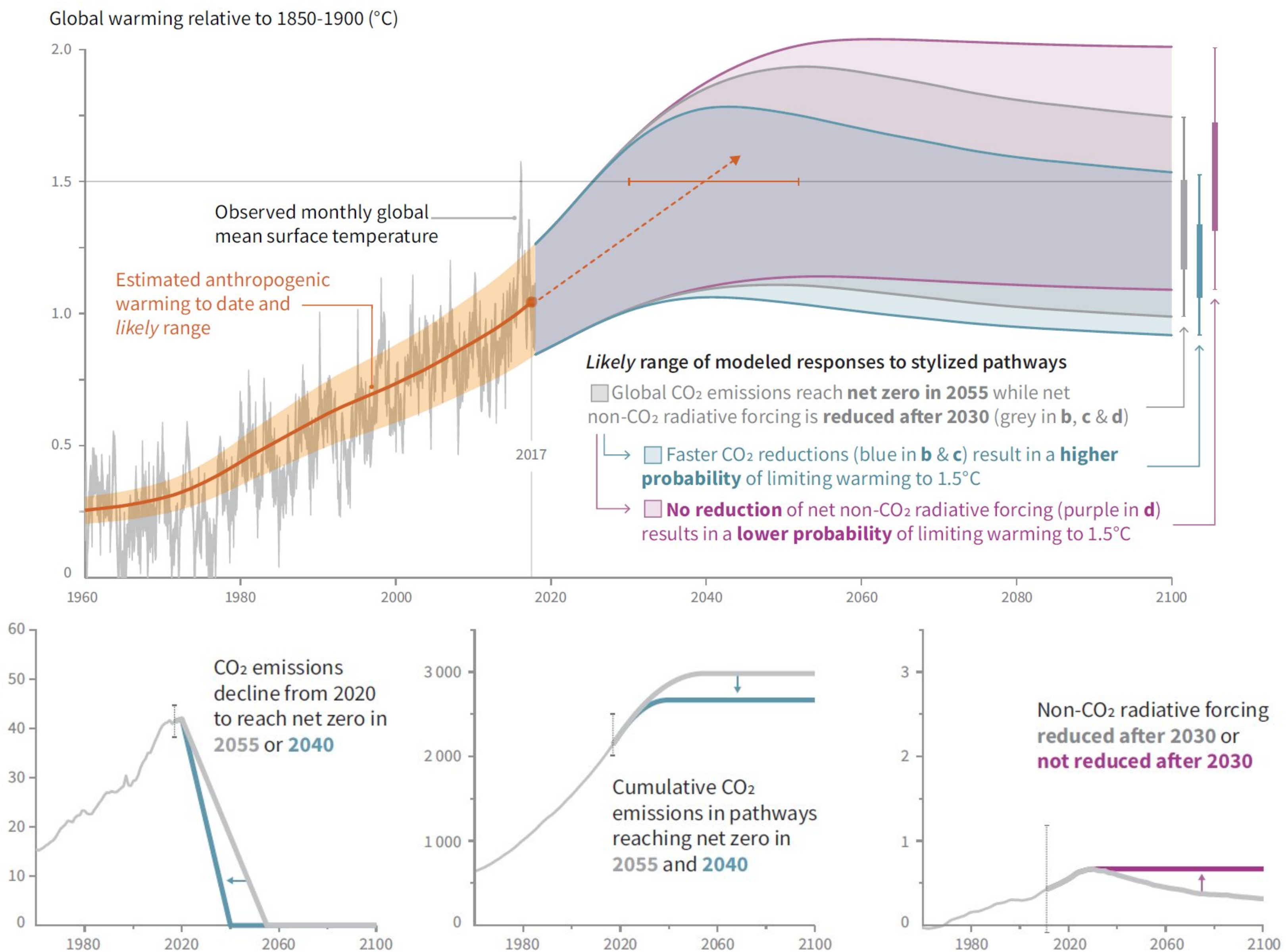
The Keeling Curve



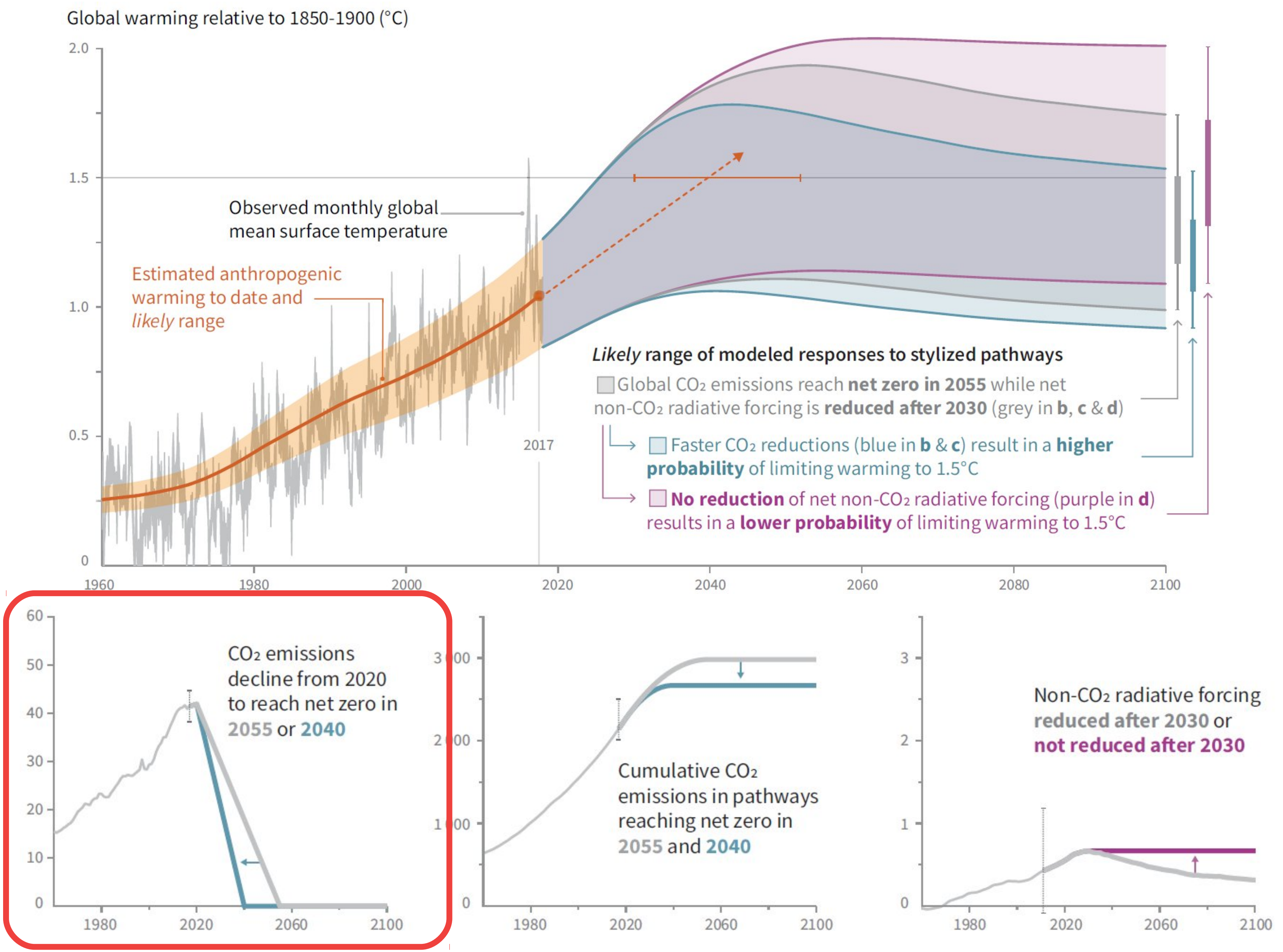
Global Temperature Rise Scenarios (IPCC 2013)



The 1.5 °C Scenario (IPCC 2018)



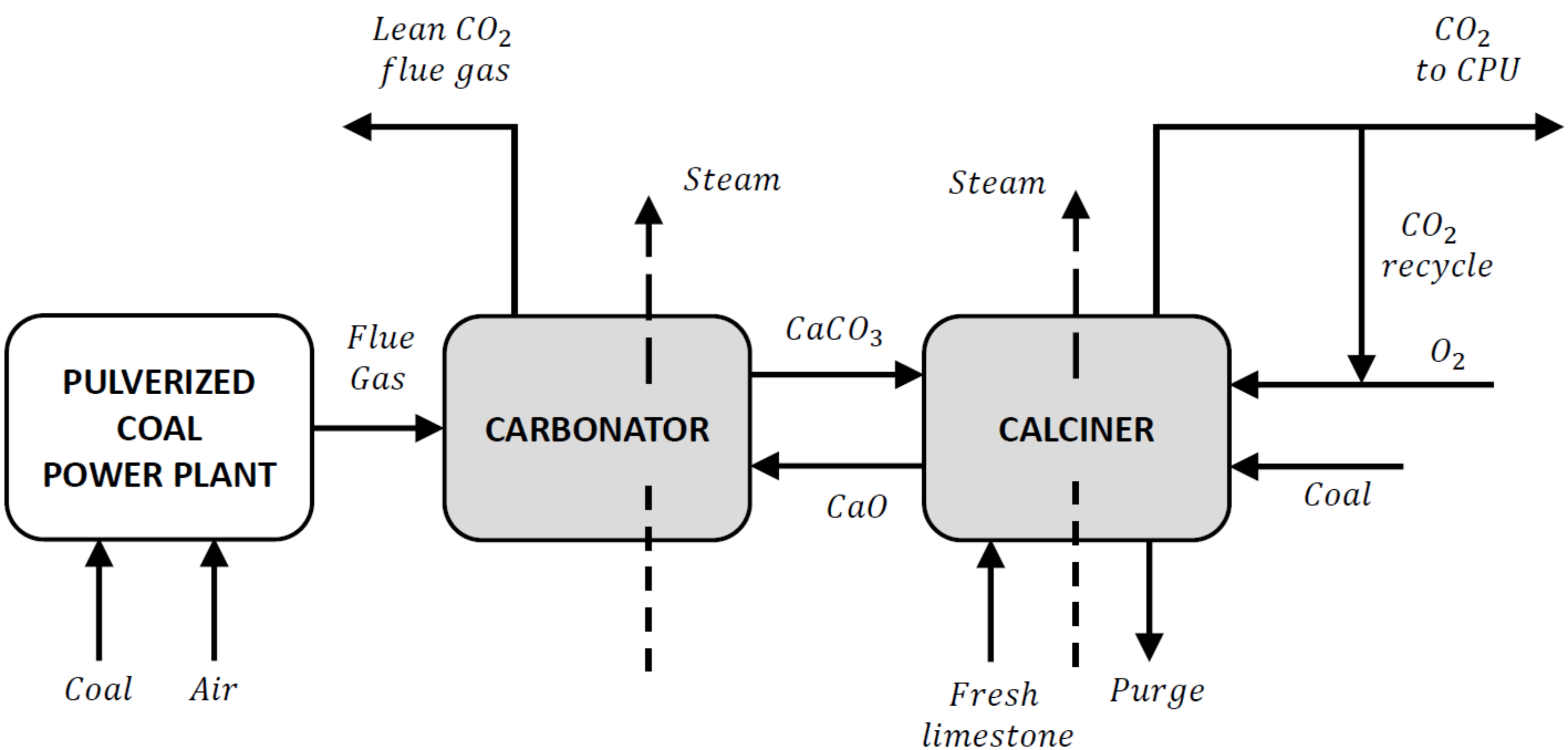
The 1.5 °C Scenario (IPCC 2018)



The FlexiCaL Project

Calcium Looping Technology (CaL)

- First proposed by Shimizu et al, 1999
- Currently demonstrated at lab and pilot scale (< 1.5 MW)



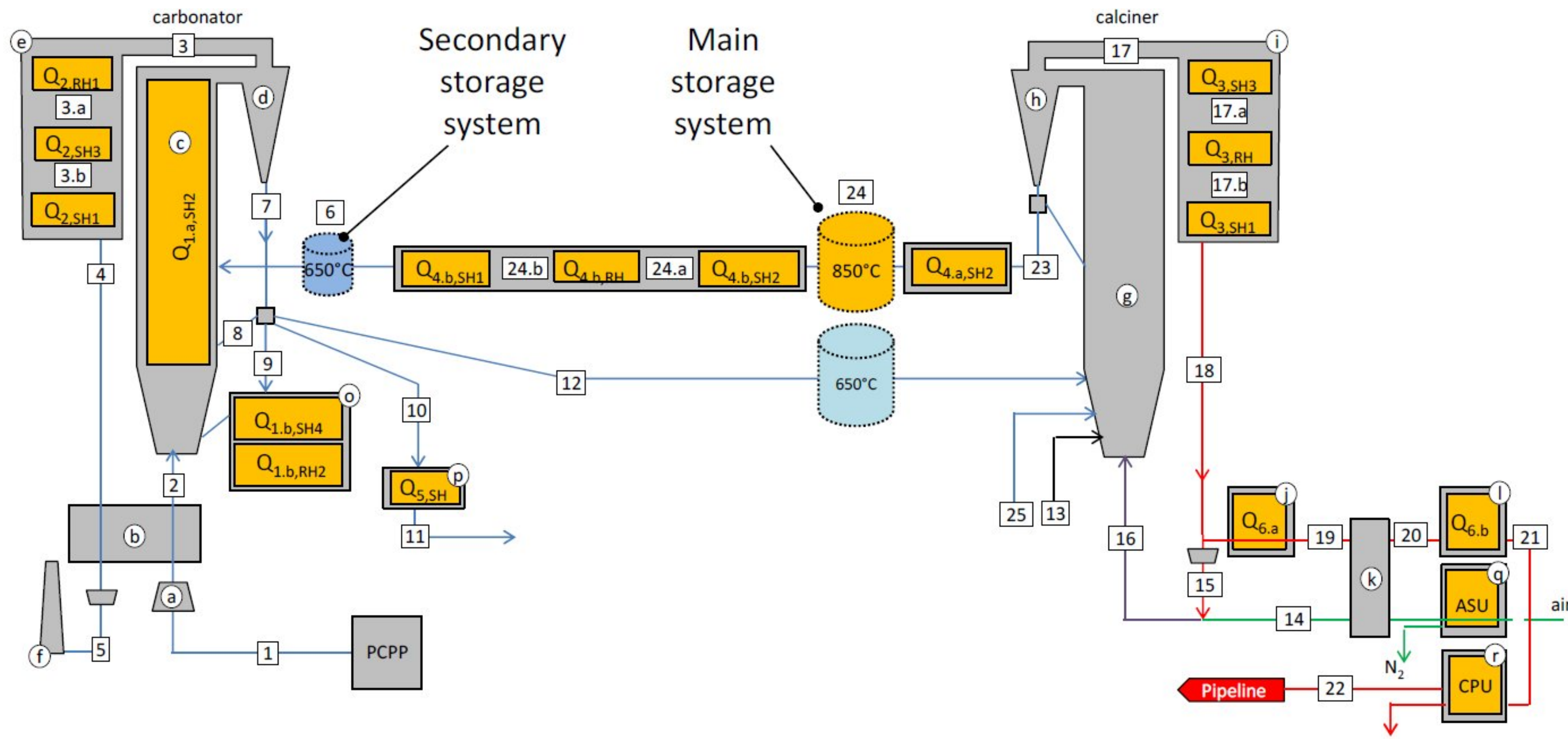
The FlexiCaL project

- Funded by the EU Research Fund for Coal and Steel (RFCS)
- Five partners from Spain, Italy, Germany, and Poland, 2016-2019

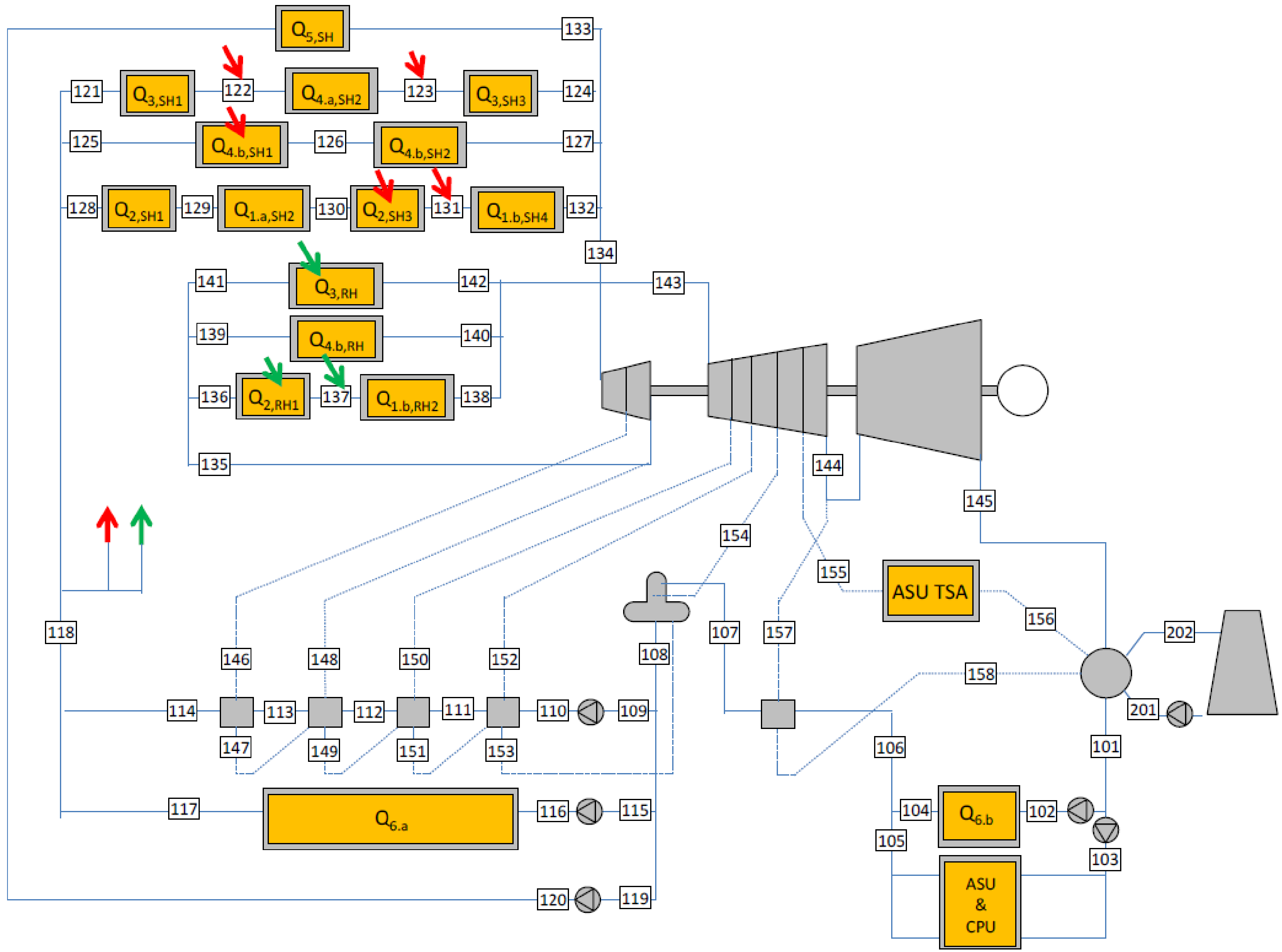
The FlexiCaL project

- Funded by the EU Research Fund for Coal and Steel (RFCS)
- Five partners from Spain, Italy, Germany, and Poland, 2016-2019
- Demonstrate flexible opertion of CaL technology at lab/pilot scale
- Design a full-scale plant capable of flexible performance
  - Off design operation
  - Transient operation

The FlexiCaL plant – CaL side



The FlexiCaL plant – Steam side



Work @ DEIB - Polimi

- Build a dynamic model of the FlexiCaL plant

- Assess the dynamic behaviour and controllability

Work @ DEIB - Polimi

- Build a dynamic model of the FlexiCaL plant
- Assess the dynamic behaviour and controllability
- Design a suitable plant-wide control strategy
- Assess the ability of the controlled FlexiCal plant to follow ramp load changes of the PCPP and to provide extra power for ancillary services

Modelling Activity

Modelling Activity @ DEIB - PoliMi

- Work in partnership DENER-DEIB (static design – dynamics and control)
- Use Modelica and re-use existing libraries as much as possible (ThermoPower and IndustrialControlSystems)

Modelling Activity @ DEIB - PoliMi

- Work in partnership DENER-DEIB (static design – dynamics and control)
- Use Modelica and re-use existing libraries as much as possible (ThermoPower and IndustrialControlSystems)
- Test individual components in steady state again design data
- Initialize full system in steady state in design conditions
- Initialize fully system in steady state in off-design conditions

Modelling Activity @ DEIB - PoliMi

- Work in partnership DENER-DEIB (static design – dynamics and control)
- Use Modelica and re-use existing libraries as much as possible (ThermoPower and IndustrialControlSystems)
- Test individual components in steady state again design data
- Initialize full system in steady state in design conditions
- Initialize fully system in steady state in off-design conditions
- Linearization of plant model around on- and off-design operating points for dynamic analysis and control design
- Design and implementation of control system in Modelica

Modelling Activity @ DEIB - PoliMi

- Work in partnership DENER-DEIB (static design – dynamics and control)
- Use Modelica and re-use existing libraries as much as possible (ThermoPower and IndustrialControlSystems)
- Test individual components in steady state again design data
- Initialize full system in steady state in design conditions
- Initialize fully system in steady state in off-design conditions
- Linearization of plant model around on- and off-design operating points for dynamic analysis and control design
- Design and implementation of control system in Modelica
- Simulation of closed loop transients
    - PCPP load ramp rate
    - CaO heat exchanger boost
    - Turbine bleed valve throttling

Modelica tools employed

- Models built and analyzed during the project using Dymola
- The model library is currently being polished in view of publication and assessed for use in OpenModelica

Modelica tools employed

- Models built and analyzed during the project using Dymola
- The model library is currently being polished in view of publication and assessed for use in OpenModelica

Perform all the modelling activities with OMC

Publish the model on github.com

Make the model usable with 100% OS toolchain

Value for OpenModelica and OSMC

- Complex, industrial-grade, numerically challenging, open-source model

Value for OpenModelica and OSMC

- Complex, industrial-grade, numerically challenging, open-source model

- Test case for OpenModelica development
  - No confidentiality issues whatsoever (100% public and paid by EU)
  - Improvement of GUI performance for editing
  - Improvement of GUI performance for post-processing (simulation results and debugging information)
  - Improvement of flattening time
  - Improvement of backend/code generation time
  - Improvement of solver robustness (particularly for initialization)
  - Improvement of simulation time

Value for OpenModelica and OSMC

- Complex, industrial-grade, numerically challenging, open-source model

- Test case for OpenModelica development
  - No confidentiality issues whatsoever (100% public and paid by EU)
  - Improvement of GUI performance for editing
  - Improvement of GUI performance for post-processing (simulation results and debugging information)
  - Improvement of flattening time
  - Improvement of backend/code generation time
  - Improvement of solver robustness (particularly for initialization)
  - Improvement of simulation time

- Showcase of the OpenModelica tool capabilities
  - Demonstration of tool performance on real-life industrial case
  - All stakeholders can download and check with their eyes, no restrictions
  - Double-edged sword!

Value for the scientific community

- Companion to forthcoming scientific publications about the FlexiCaL plant dynamic modelling and control (publish the paper **and** the model)

Value for the scientific community

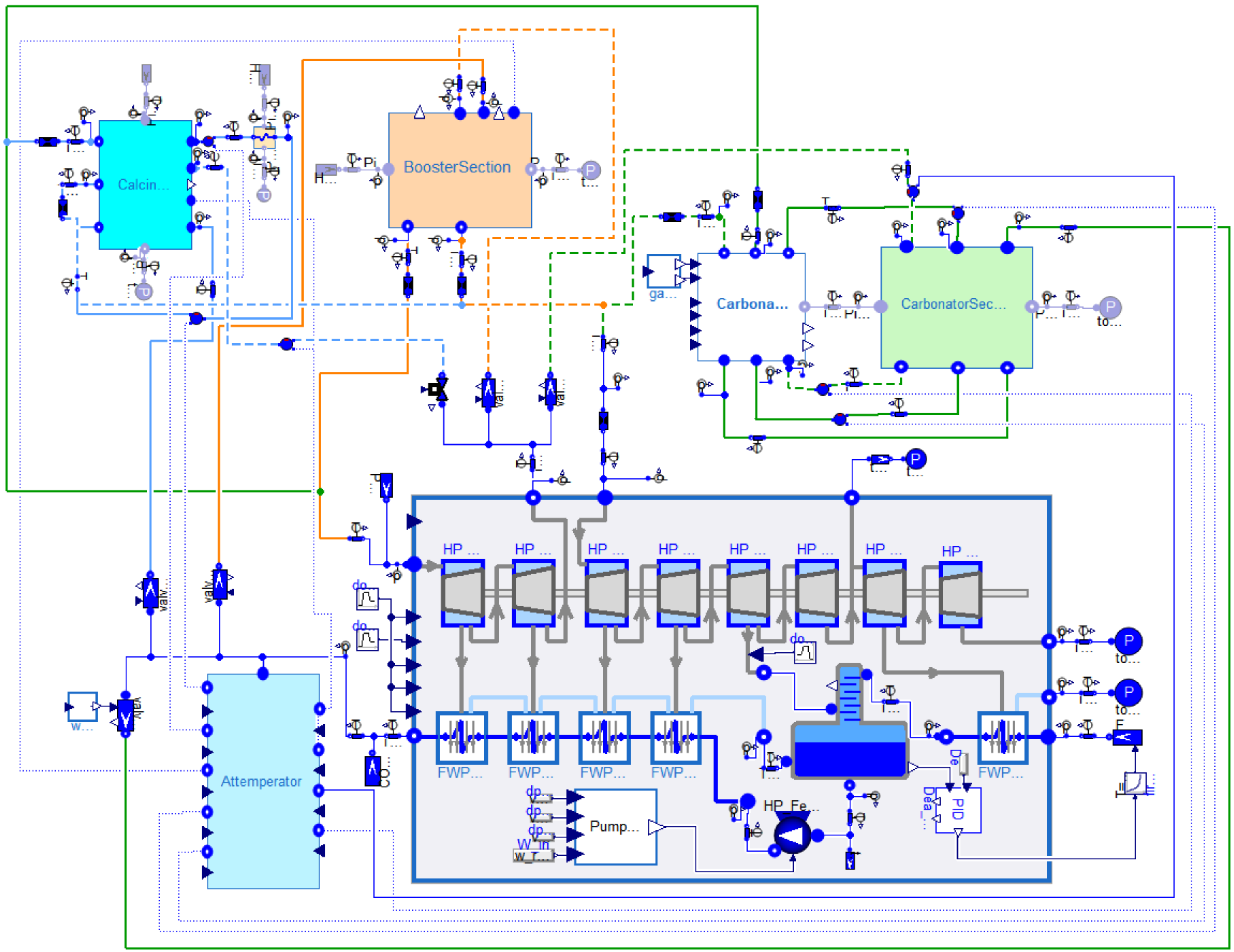
- Companion to forthcoming scientific publications about the FlexiCaL plant dynamic modelling and control (publish the paper **and** the model)
- Allows to reproduce the published results with no license restrictions
- Allows other people to reuse the model for further research

Value for the scientific community

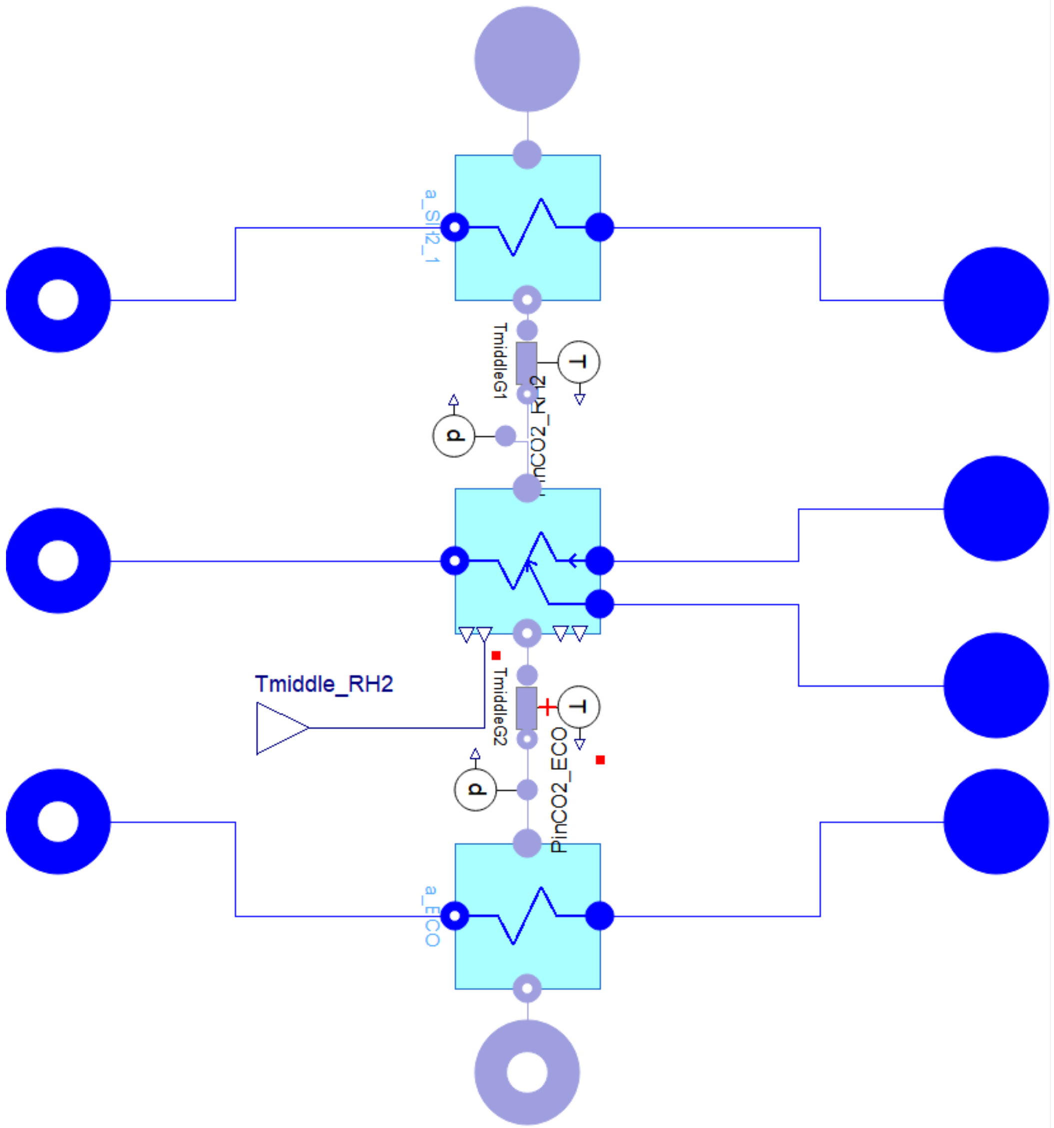
- Companion to forthcoming scientific publications about the FlexiCaL plant dynamic modelling and control (publish the paper **and** the model)
- Allows to reproduce the published results with no license restrictions
- Allows other people to reuse the model for further research
- Provides engineers and researchers with a fully developed example
  - How to organize a complex power plant model in Modelica
  - How to carry out different activities in an efficient way without code duplication
  - Significant tutorial value

Presentation of the Model

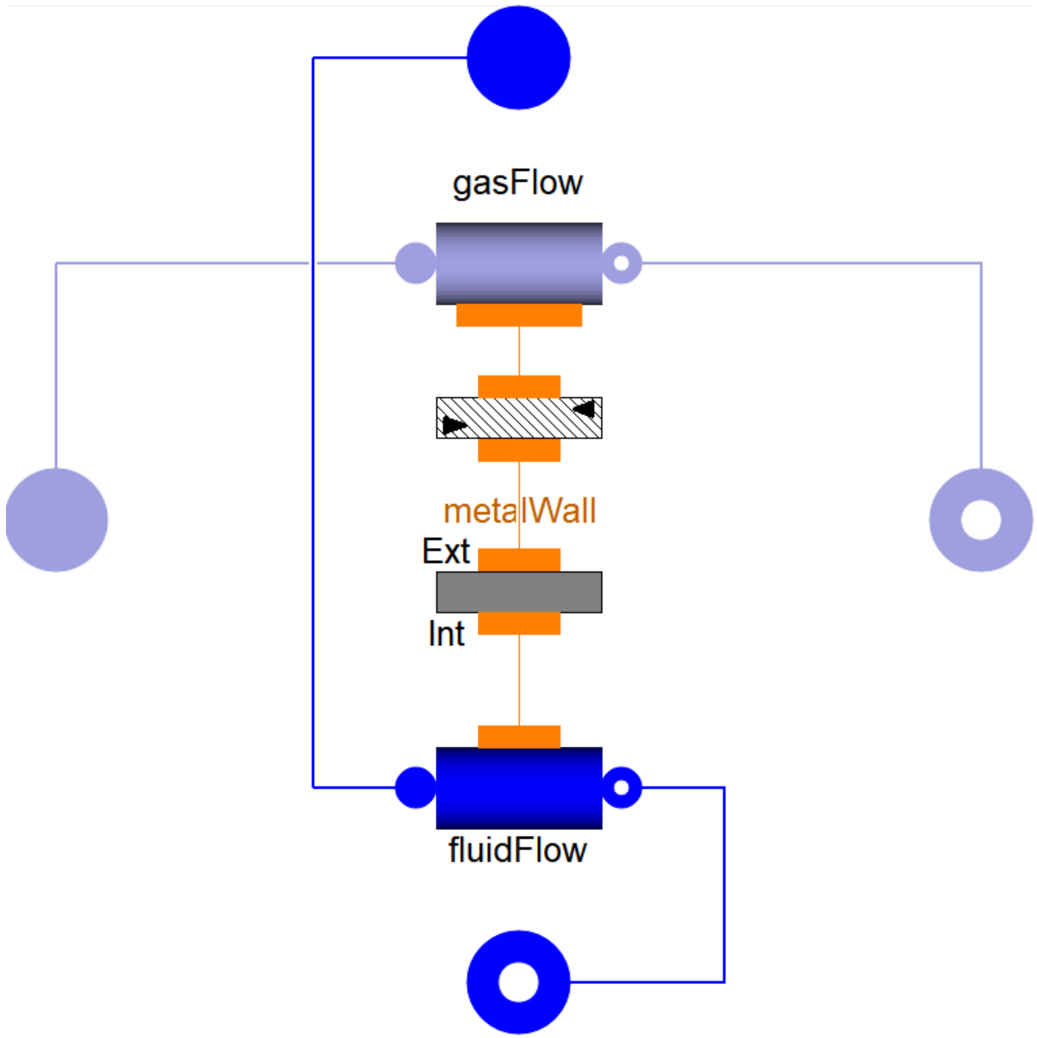
Top-level view of the plant model



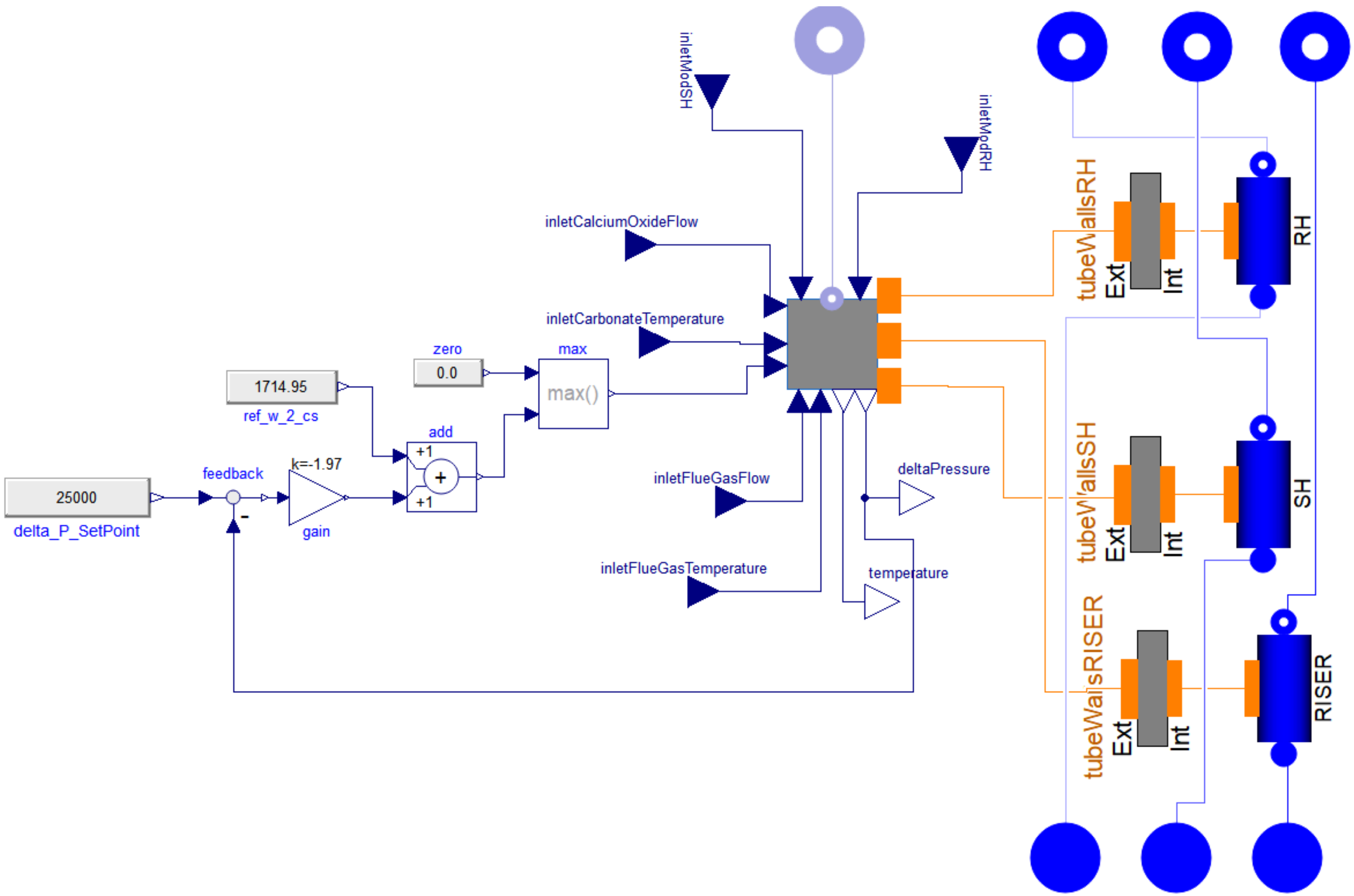
Calciner exhaust section



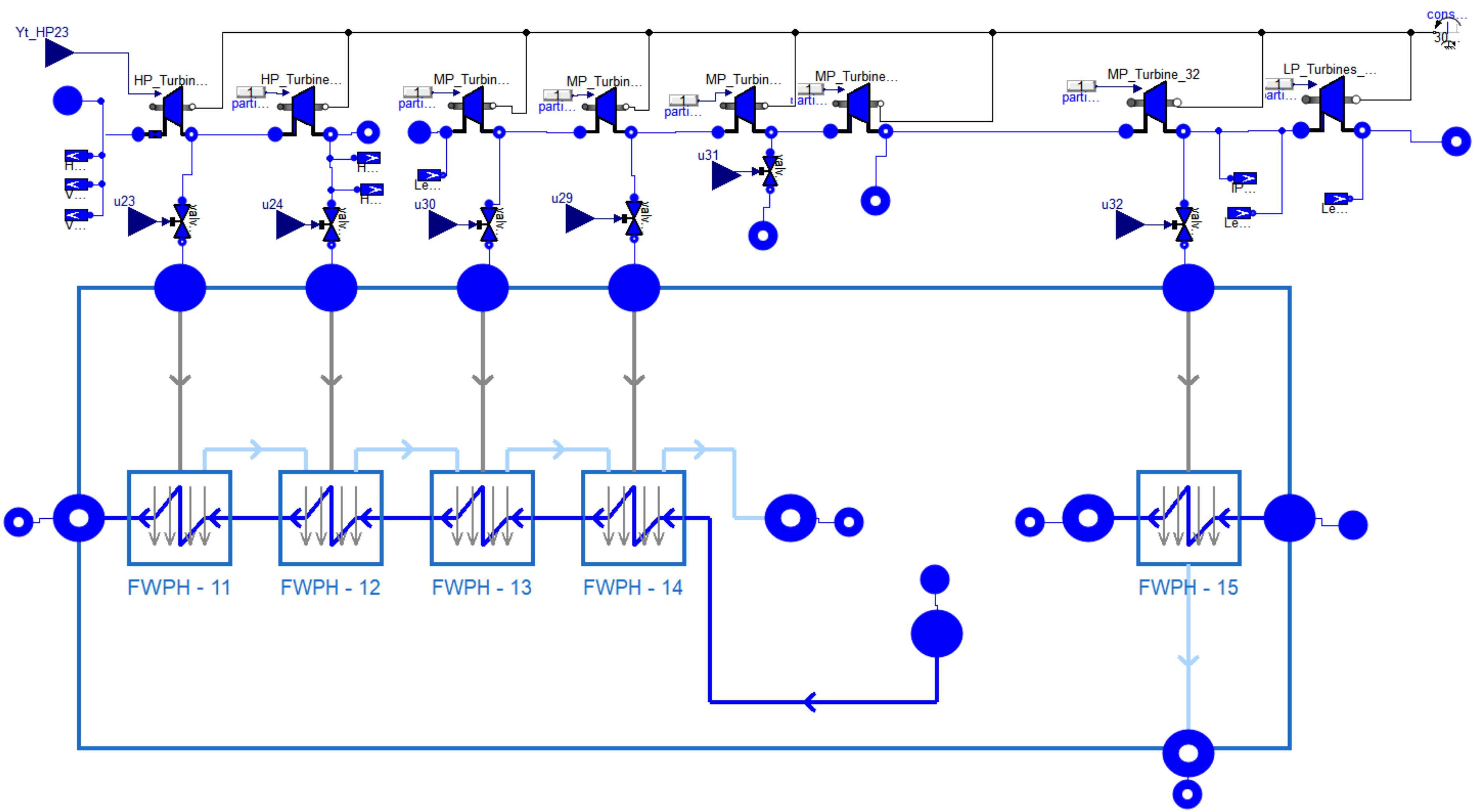
Individual steam – fluidized CaO heat exchanger



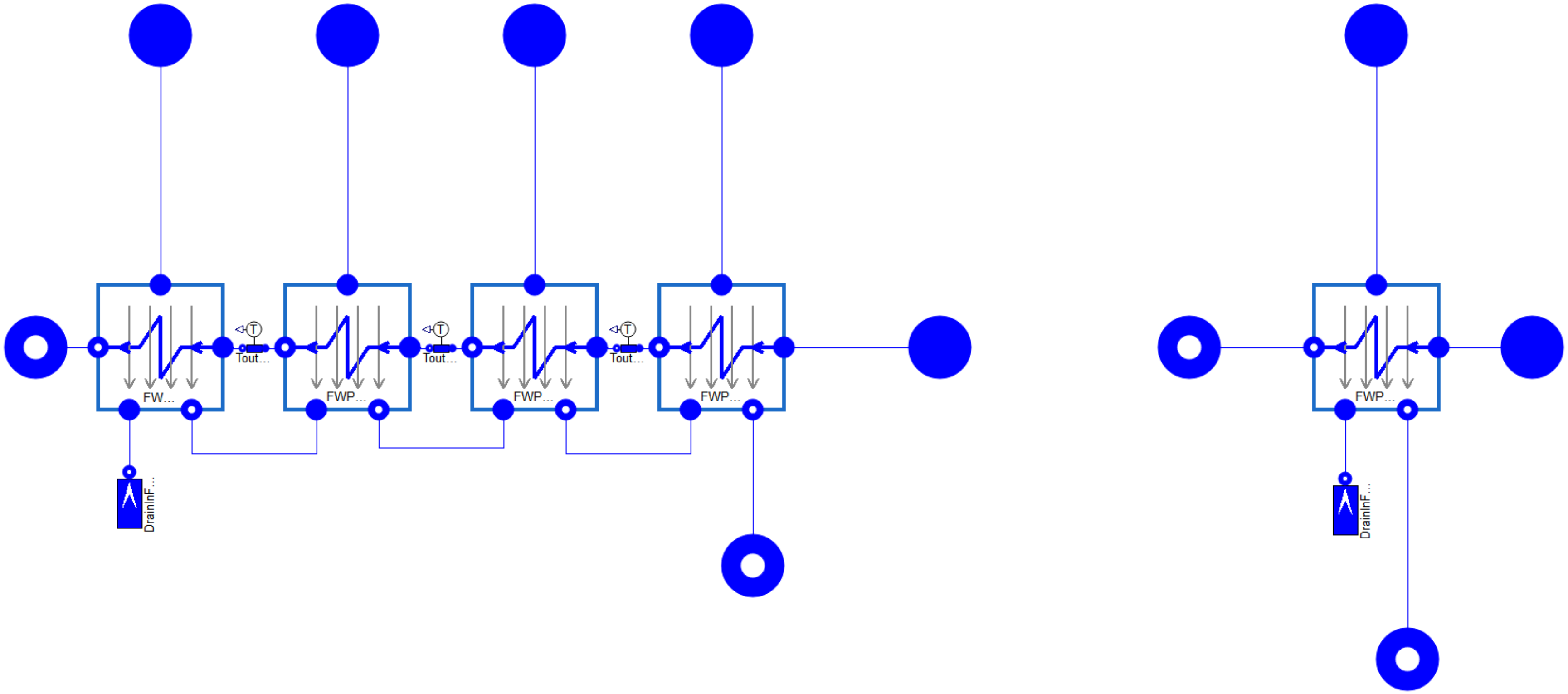
Carbonator model



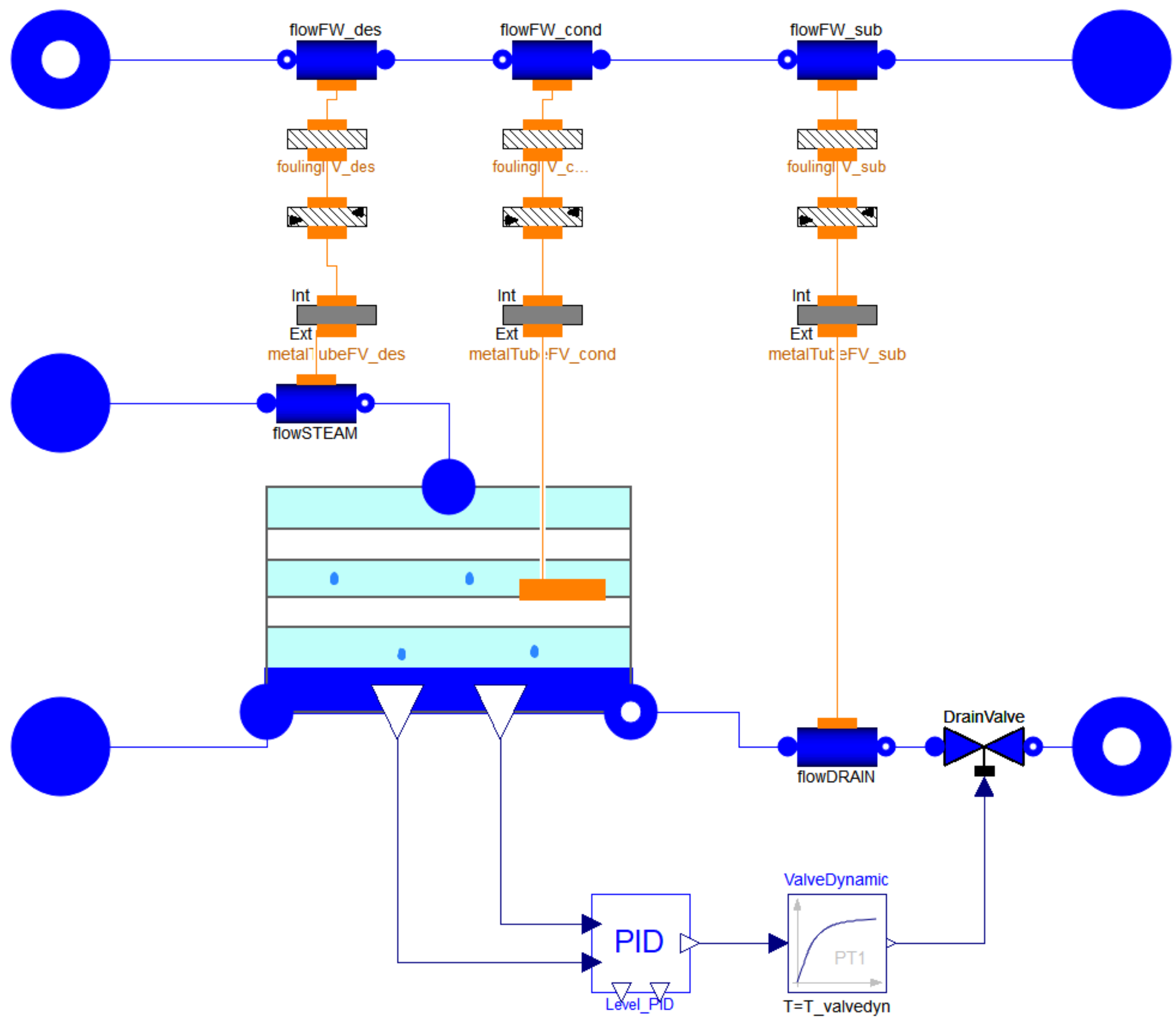
Turbine – Feedwater train model



Accurate feedwater train model



Accurate water preheater model



Steady-state initialization modes

- Forward on-design steady-state
  - Inputs are fixed to design values
  - Initial equations with zero derivatives on all states

Steady-state initialization modes

- Forward on-design steady-state
  - Inputs are fixed to design values
  - Initial equations with zero derivatives on all states
- Backward on-design steady-state
  - Outputs are fixed to design values in initial equations
  - Inputs assigned to fixed = false parameters
  - Initial equations with zero derivatives on all states

Steady-state initialization modes

- Forward on-design steady-state
  - Inputs are fixed to design values
  - Initial equations with zero derivatives on all states
- Backward on-design steady-state
  - Outputs are fixed to design values in initial equations
  - Inputs assigned to fixed = false parameters
  - Initial equations with zero derivatives on all states
- Backward off-design steady-state
  - Outputs are fixed to off-design values in initial equations, as functions of a load value
  - Inputs assigned to fixed = false parameters
  - Initial equations with zero derivatives on all states
  - Homotopy brings load gradually from 100% to required value

Steady-state initialization modes

- Forward on-design steady-state
  - Inputs are fixed to design values
  - Initial equations with zero derivatives on all states
- Backward on-design steady-state
  - Outputs are fixed to design values in initial equations
  - Inputs assigned to fixed = false parameters
  - Initial equations with zero derivatives on all states
- Backward off-design steady-state
  - Outputs are fixed to off-design values in initial equations, as functions of a load value
  - Inputs assigned to fixed = false parameters
  - Initial equations with zero derivatives on all states
  - Homotopy brings load gradually from 100% to required value
- Backward off-design steady-state for linearization and open-loop step response computation
  - As above, but top level inputs and outputs are added

Steady-state initialization modes

- Forward on-design steady-state
  - Inputs are fixed to design values
  - Initial equations with zero derivatives on all states
- Backward on-design steady-state
  - Outputs are fixed to design values in initial equations
  - Inputs assigned to fixed = false parameters
  - Initial equations with zero derivatives on all states
- Backward off-design steady-state
  - Outputs are fixed to off-design values in initial equations, as functions of a load value
  - Inputs assigned to fixed = false parameters
  - Initial equations with zero derivatives
  - Homotopy brings load gradually from 100% to required value
- Backward off-design steady-state for linearization and open-loop step response computation
  - As above, but top level inputs and outputs are added
- Closed-loop steady state
  - Control system is added, with given set points and load level
  - Suitable homotopy introduced on controllers to facilitate convergence

Current Status with OpenModelica 1.16.0-dev

Testing setup

- Lenovo Carbon X1
- CPU: Intel i7-8550, 1.8 Ghz, 8 virtual cores
- RAM: 16 GB
- SSD: 1 TB
- OS: Windows 10 Professional 64 bit
- OMC: 1.16.0-dev nighly build of 31 Jan 2020

GUI

- The diagrams are mostly displayed correctly
- Some glitches with extent
- Editing the models with few components is fine
- Parameter input is fine
- Replaceable Medium missing ($\rightarrow$ v. 1.15.0)
- Editing the top model is still too slow (20 s response time)

GUI

- The diagrams are mostly displayed correctly
- Some glitches with extent
- Editing the models with few components is fine
- Parameter input is fine
- Replaceable Medium missing (→ v. 1.15.0)
- Editing the top model is still too slow (20 s response time)

Continue improvement of API used by OMEdit by using the faster new frontend

Performance of full plant model

- Model size: 19325 equations (9244 trivial)
- Flattening time using new front end: 16 s (Dymola: 8 s)

Performance of full plant model

- Model size: 19325 equations (9244 trivial)
- Flattening time using new front end: 16 s (Dymola: 8 s)
- Backend + code generation: 3 m
  - Matching and sorting: 10 s
  - Analyze initial system: 18 s
  - Tearing init system: 36 s
  - Tearing simplified init system: 23 s
  - Remove simple equations: 7.5 s
  - Templates: 27 s

Performance of full plant model

- Model size: 19325 equations (9244 trivial)
- Flattening time using new front end: 16 s (Dymola: 8 s)
- Backend + code generation: 3 m
  - Matching and sorting: 10 s
  - Analyze initial system: 18 s
  - Tearing init system: 36 s
  - Tearing simplified init system: 23 s
  - Remove simple equations: 7.5 s
  - Templates: 27 s
- C-code compilation (gcc): 4 m
- Total compilation time: 7 m 40 s (Dymola: 23 s)
- Max memory usage: 12 GB
- Initialization fails due to wrongly selected initial guesses (issue currently under investigation)

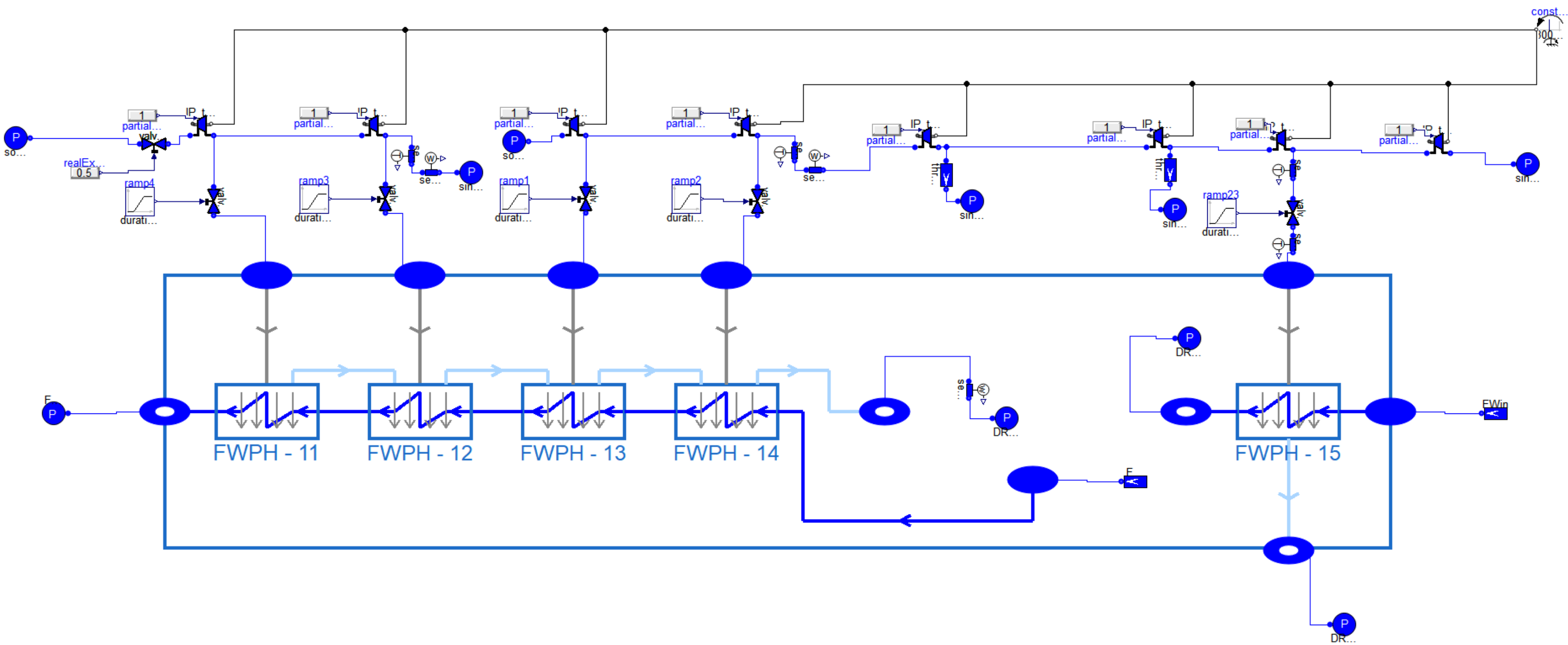
Performance of full plant model

- Model size: 19325 equations (9244 trivial)
- Flattening time using new front end: 16 s (Dymola: 8 s)
- Backend + code generation: 3 m
  - Matching and sorting: 10 s
  - Analyze initial system: 18 s
  - Tearing init system: 36 s
  - Tearing simplified init system: 23 s
  - Remove simple equations: 7.5 s
  - Templates: 27 s
- C-code compilation (gcc): 4 m
- Total compilation time: 7 m 40 s (Dymola: 23 s)
- Max memory usage: 12 GB
- Initialization fails due to wrongly selected initial guesses (issue currently under investigation)

Room for improvement in code generation

- Faster selected back-end methods and templates
- Faster compilation (gcc → clang, factor 5)
- Resolve initial guess issues

Smaller case study: Feedwater/turbine unit test



Performance of turbine/feedwater unit test

- Model size: 8650 equations (3731 trivial)
- Flattening time using new front end: 4 s (Dymola: 2.5 s)
- Backend + code generation: 36 s (Dymola: 10 s)
- C-code compilation (gcc): 1 m (Dymola: 6 s)
- Total compilation time: 1 m 40 s (Dymola: 18 s)
- Max memory usage: 3.7 GB
- Initialization time: 70 s (Dymola: 2.5 s)
- Simulation time: 6.5 s (Dymola: 4.2 s)

Performance of turbine/feedwater unit test

- Model size: 8650 equations (3731 trivial)
- Flattening time using new front end: 4 s (Dymola: 2.5 s)
- Backend + code generation: 36 s (Dymola: 10 s)
- C-code compilation (gcc): 1 m (Dymola: 6 s)
- Total compilation time: 1 m 40 s (Dymola: 18 s)
- Max memory usage: 3.7 GB
- Initialization time: 70 s (Dymola: 2.5 s)
- Simulation time: 6.5 s (Dymola: 4.2 s)

Room for improvement at runtime:

- Performe CSE also during init (WiP)
- Improve optimization of Modelica.Media IF97 code

Outlook
&
Conclusions

Outlook

- Completion of the FlexiCaL model clean up: Feb-Mar 2020
- Model published on github.com afterwards
- Will be included in the OpenModelica library testuite

Outlook

- Completion of the FlexiCaL model clean up: Feb-Mar 2020
- Model published on github.com afterwards
- Will be included in the OpenModelica library testuite
- Paradigmatic case for the assessment and improvement of OpenModelica performance in a tough industrial case
  - Editing by the OMEdit GUI
  - Flattening time
  - Code generation time
  - C-code compilation time
  - Initialization time
  - Simulation time

Outlook

- Completion of the FlexiCaL model clean up: Feb-Mar 2020
- Model published on github.com afterwards
- Will be included in the OpenModelica library testuite
- Paradigmatic case for the assessment and improvement of OpenModelica performance in a tough industrial case
  - Editing by the OMEdit GUI
  - Flattening time
  - Code generation time
  - C-code compilation time
  - Initialization time
  - Simulation time

⇨ Opportunity to improve the quality of OpenModelica for industrial users

Conclusions

- The RFCS FlexiCaL project aimed at designing a novel CCS plant
- Flexible performance from a control viewpoint was studied
- Modelica models of the FlexiCaL plant are publicly available

Conclusions

- The RFCS FlexiCaL project aimed at designing a novel CCS plant
- Flexible performance from a control viewpoint was studied
- Modelica models of the FlexiCaL plant are publicly available
- Uses:
  - Companion of scientific papers, full disclosureù and reproducibility of the results
  - Benchmark for OMC performance improvement
  - Showcase for serious industrial use of OMC
- OpenModelica allows to run them and share them w/o licensing issue
- OMC 1.16.0 (release date June 2020) will be up to the task

Conclusions

- The RFCS FlexiCaL project aimed at designing a novel CCS plant
- Flexible performance from a control viewpoint was studied
- Modelica models of the FlexiCaL plant are publicly available
- Uses:
  - Companion of scientific papers, full disclosureù and reproducibility of the results
  - Benchmark for OMC performance improvement
  - Showcase for serious industrial use of OMC
- OpenModelica allows to run them and share them w/o licensing issue
- OMC 1.16.0 (release date June 2020) will be up to the task
- On going work
  - Streamlining OMEdit editor response for larger models (nfAPI)
  - Further speed-up of flattening
  - Optimization and speed-up of backend
  - Improve CSE handling in the runtime, part. at initialization

Thank you for your kind attention!