

The *Dymola* Thermo-Bond-Graph Library

- In this lecture, we shall introduce a third bond-graph library, one designed explicitly to deal with *convective flows*.
- To this end, we shall need to introduce a new type of bonds, bonds carrying in parallel three distinct, yet inseparable, power flows: a *heat flow*, a *volume flow*, and a *mass flow*.
- These new *bus-bonds*, together with their corresponding *bus-0-junctions*, enable the modeler to describe convective flows at a high abstraction level.
- The example of a *pressure cooker* model completes the presentation.

Table of Contents

- Thermo-bond graph connectors
- A-causal and causal bonds
- Bus-junctions
- Heat exchanger
- Volume work
- Forced volume flow
- Resistive field
- Pressure cooker
- Capacitive fields
- Evaporation and condensation
- Simulation of pressure cooker
- Free convective mass flow
- Free convective volume flow
- Forced convective volume flow
- Water serpentine
- Biosphere 2

The Thermo-Bond Graph Connectors I

- We shall need to introduce new *thermo-bond graph connectors* to carry the six variables associated with the three flows. They are designed as an *11-tuple*.

```

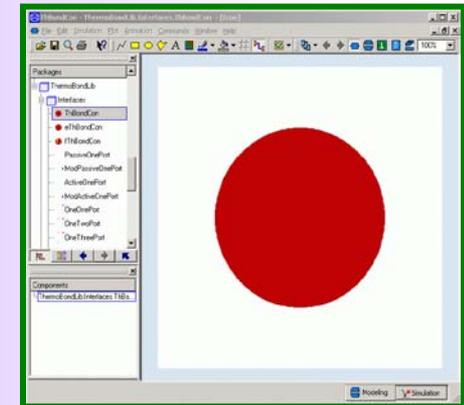
connector ThBondCon "Bi-directional thermo-bond graph connector"
  Modelica.SIunits.Temperature T "Temperature";
  Modelica.SIunits.Pressure p "Pressure";
  Modelica.SIunits.SpecificEnthalpy g "Gibbs potential";
  Modelica.SIunits.ThermalConductance Sdot "Entropy flow";
  Modelica.SIunits.VolumeFlowRate q "Volume flow";
  Modelica.SIunits.MassFlowRate Mdot "Mass flow";
  Modelica.SIunits.Entropy S "Entropy";
  Modelica.SIunits.Volume V "Volume";
  Modelica.SIunits.Mass M "Mass";
  Real d "Directional variable";
  Boolean Exist "True if substance exists";
end ThBondCon;
  
```

} efforts, e

} flows, f

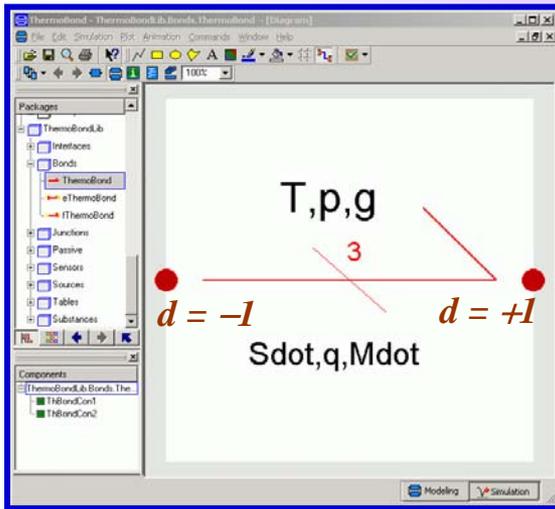
} generalized positions, q

directional variable, d
indicator variable



The thermo-bond connector icon is a red dot.

The A-Causal Thermo-Bond Model



*The thermo-bond is equivalent to the regular bond, except that **ten** variables need to be connected across, instead of only **two**.*

Screenshot of the Modelica IDE showing the ThermoBond model definition. The Packages pane shows ThermoBondLib, Interfaces, Bonds, and ThermoBond. The Modelica Text pane shows the model definition with the equation section containing `ThBondCon2.d = -1;` and `ThBondCon2.d = +1;`.

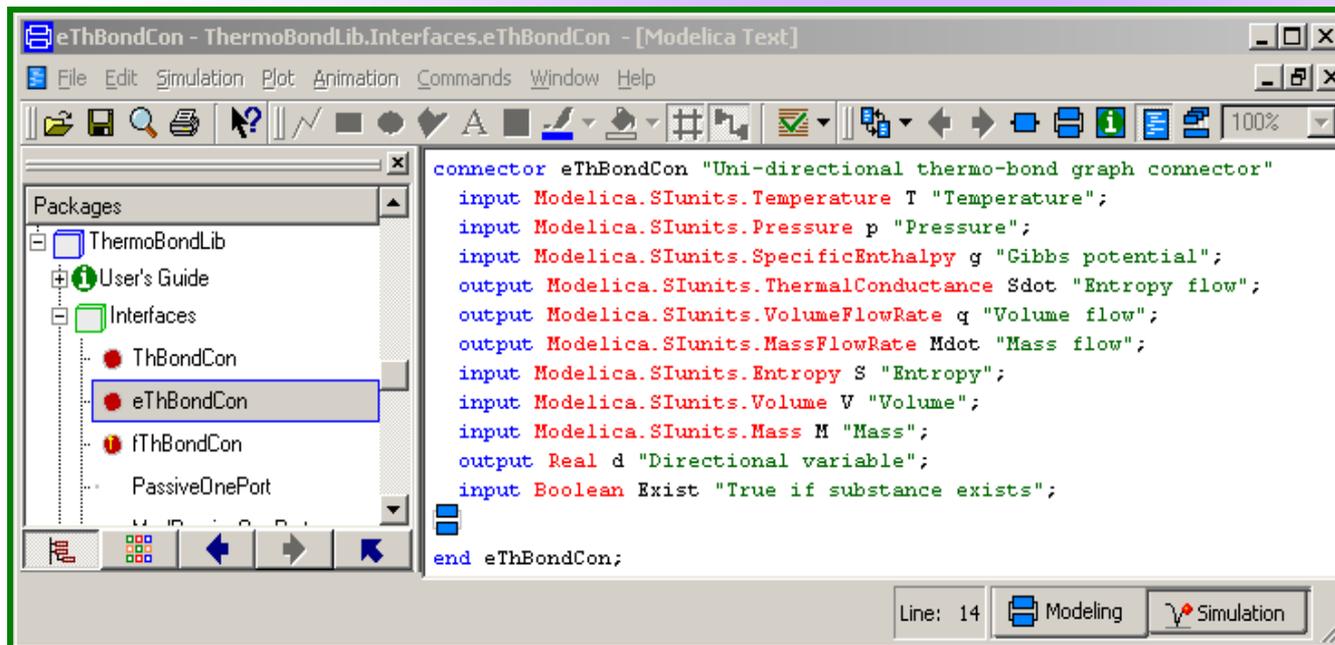
```

model ThermoBond
equation
  ThBondCon2.T = ThBondCon1.T;
  ThBondCon2.p = ThBondCon1.p;
  ThBondCon2.g = ThBondCon1.g;
  ThBondCon2.Sdot = ThBondCon1.Sdot;
  ThBondCon2.q = ThBondCon1.q;
  ThBondCon2.Mdot = ThBondCon1.Mdot;
  ThBondCon2.S = ThBondCon1.S;
  ThBondCon2.V = ThBondCon1.V;
  ThBondCon2.M = ThBondCon1.M;
  ThBondCon2.Exist = ThBondCon1.Exist;
  ThBondCon1.d = -1;
  ThBondCon2.d = +1;
end ThermoBond;

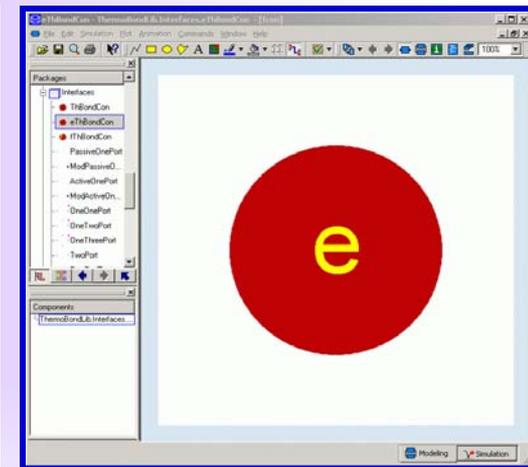
```

The Thermo-Bond Graph Connectors II

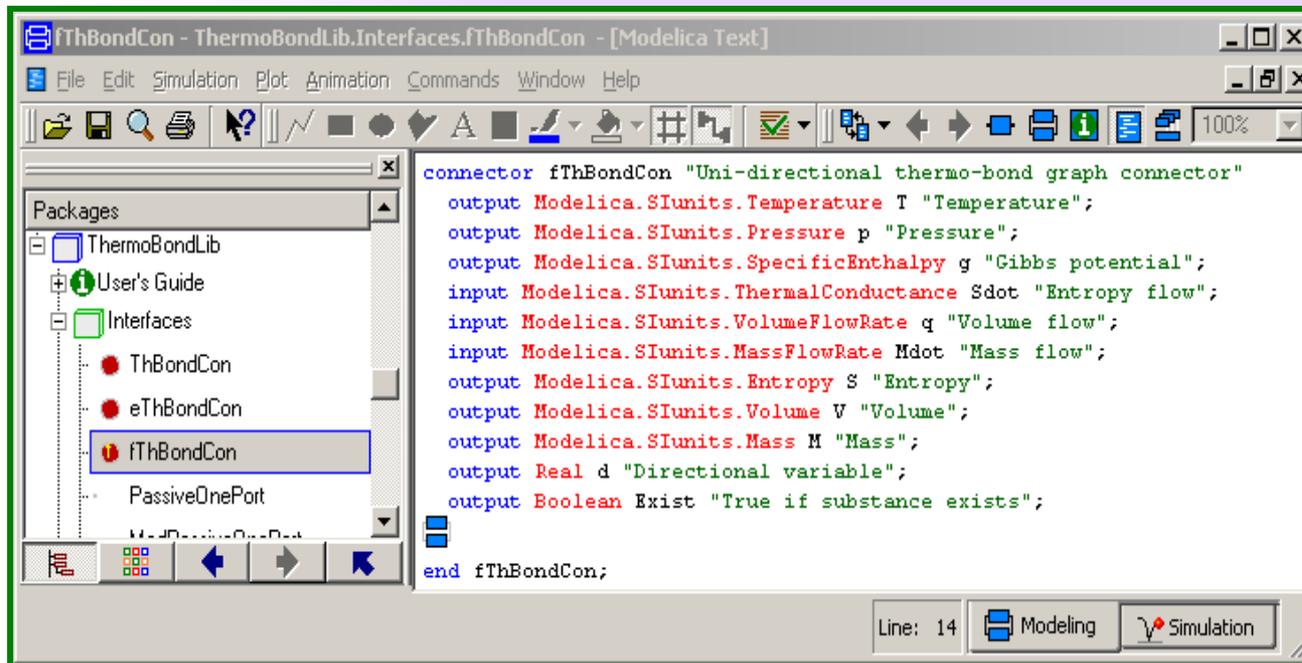
- Like in the case of the general bond-graph library, also the thermo-bond-graph library offers *causal* next to *a-causal* bonds.



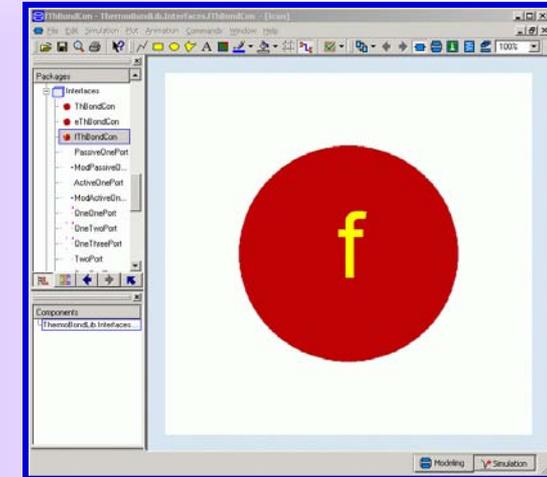
```
connector eThBondCon "Uni-directional thermo-bond graph connector"  
  input Modelica.SIunits.Temperature T "Temperature";  
  input Modelica.SIunits.Pressure p "Pressure";  
  input Modelica.SIunits.SpecificEnthalpy g "Gibbs potential";  
  output Modelica.SIunits.ThermalConductance Sdot "Entropy flow";  
  output Modelica.SIunits.VolumeFlowRate q "Volume flow";  
  output Modelica.SIunits.MassFlowRate Mdot "Mass flow";  
  input Modelica.SIunits.Entropy S "Entropy";  
  input Modelica.SIunits.Volume V "Volume";  
  input Modelica.SIunits.Mass M "Mass";  
  output Real d "Directional variable";  
  input Boolean Exist "True if substance exists";  
end eThBondCon;
```



The Thermo-Bond Graph Connectors III

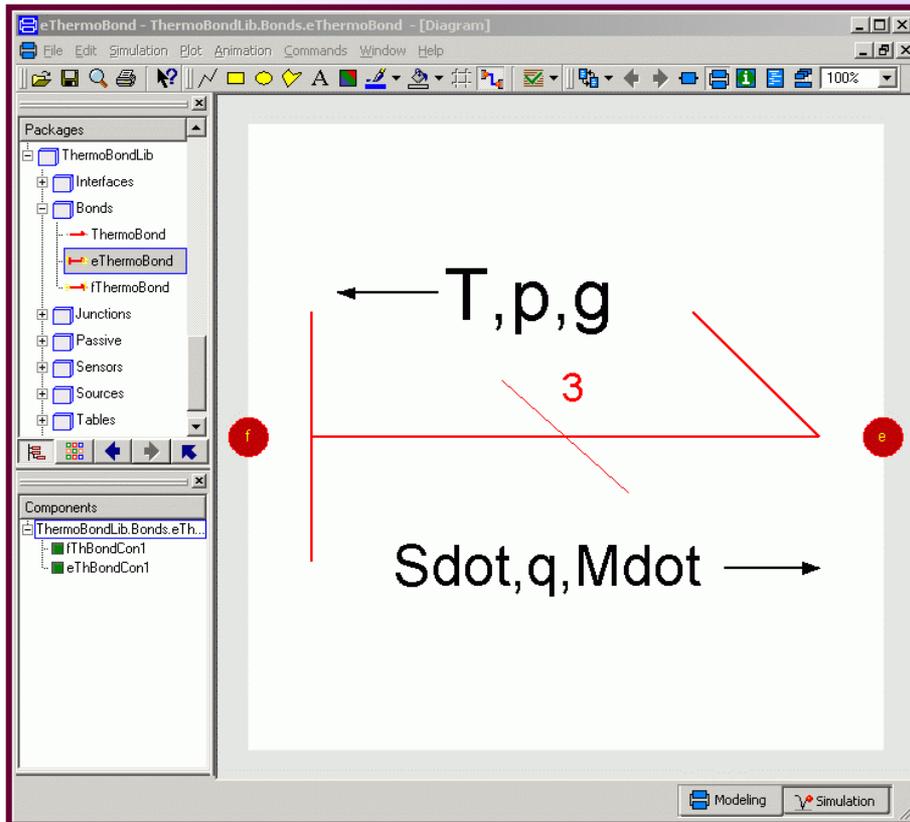


```
connector fThBondCon "Uni-directional thermo-bond graph connector"
output Modelica.SIunits.Temperature T "Temperature";
output Modelica.SIunits.Pressure p "Pressure";
output Modelica.SIunits.SpecificEnthalpy g "Gibbs potential";
input Modelica.SIunits.ThermalConductance Sdot "Entropy flow";
input Modelica.SIunits.VolumeFlowRate q "Volume flow";
input Modelica.SIunits.MassFlowRate Mdot "Mass flow";
output Modelica.SIunits.Entropy S "Entropy";
output Modelica.SIunits.Volume V "Volume";
output Modelica.SIunits.Mass M "Mass";
output Real d "Directional variable";
output Boolean Exist "True if substance exists";
end fThBondCon;
```



- Either the three efforts or the three flows are treated as input variables.

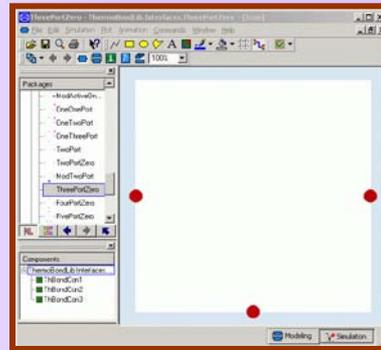
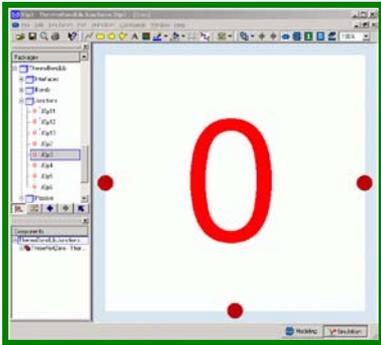
The Causal Thermo-Bond Blocks



- Using these connectors, causal thermo-bond blocks can be defined.
- The *f-connector* is used at the side of the causality stroke.
- The *e-connector* is used at the other side.
- The causal thermo-bond-graph connectors are only used in the context of the thermo-bond blocks. Everywhere else, the a-causal thermo-bond-graph connectors are to be used.

The Bus-0-Junctions

The junctions can now be programmed. Let us look at a *bus-0-junction with three bond attachments*.



Modelica Text Editor: J0p3 - ThermoBondLib.Junctions.J0p3

```

model J0p3
  extends Interfaces.ThreePortZero;
equation
  T[2:3] = T[1:2];
  p[2:3] = p[1:2];
  g[2:3] = g[1:2];
  sum(Sdot) = 0;
  sum(q) = 0;
  sum(Mdot) = 0;
  S[2:3] = S[1:2];
  V[2:3] = V[1:2];
  M[2:3] = M[1:2];
  Exist[2:3] = Exist[1:2];
end J0p3;
  
```

Line: 1 Modeling Simulation

Modelica Text Editor: ThreePortZero - ThermoBondLib.Interfaces.ThreePortZero

```

model ThreePortZero
  "Partial model invoking three thermo-bond graph connectors"
  Modelica.SIunits.Temperature T[3] "Temperature";
  Modelica.SIunits.Pressure p[3] "Pressure";
  Modelica.SIunits.SpecificEnthalpy g[3] "Gibbs potential";
  Modelica.SIunits.ThermalConductance Sdot[3] "Entropy flow";
  Modelica.SIunits.VolumeFlowRate q[3] "Volume flow";
  Modelica.SIunits.MassFlowRate Mdot[3] "Mass flow";
  Modelica.SIunits.Entropy S[3] "Entropy";
  Modelica.SIunits.Volume V[3] "Volume";
  Modelica.SIunits.Mass M[3] "Mass";
  Boolean Exist[3] "True if substance exists";
equation
  T[1] = ThBondCon1.T;
  p[1] = ThBondCon1.p;
  g[1] = ThBondCon1.g;
  Sdot[1] = ThBondCon1.d*ThBondCon1.Sdot;
  q[1] = ThBondCon1.d*ThBondCon1.q;
  Mdot[1] = ThBondCon1.d*ThBondCon1.Mdot;
  S[1] = ThBondCon1.S;
  V[1] = ThBondCon1.V;
  M[1] = ThBondCon1.M;
  Exist[1] = ThBondCon1.Exist;
  T[2] = ThBondCon2.T;
  p[2] = ThBondCon2.p;
  g[2] = ThBondCon2.g;
  Sdot[2] = ThBondCon2.d*ThBondCon2.Sdot;
  q[2] = ThBondCon2.d*ThBondCon2.q;
  Mdot[2] = ThBondCon2.d*ThBondCon2.Mdot;
  S[2] = ThBondCon2.S;
  V[2] = ThBondCon2.V;
  M[2] = ThBondCon2.M;
  Exist[2] = ThBondCon2.Exist;
  T[3] = ThBondCon3.T;
  p[3] = ThBondCon3.p;
  g[3] = ThBondCon3.g;
  Sdot[3] = ThBondCon3.d*ThBondCon3.Sdot;
  q[3] = ThBondCon3.d*ThBondCon3.q;
  Mdot[3] = ThBondCon3.d*ThBondCon3.Mdot;
  S[3] = ThBondCon3.S;
  V[3] = ThBondCon3.V;
  M[3] = ThBondCon3.M;
  Exist[3] = ThBondCon3.Exist;
  
```

Line: 3 Modeling Simulation

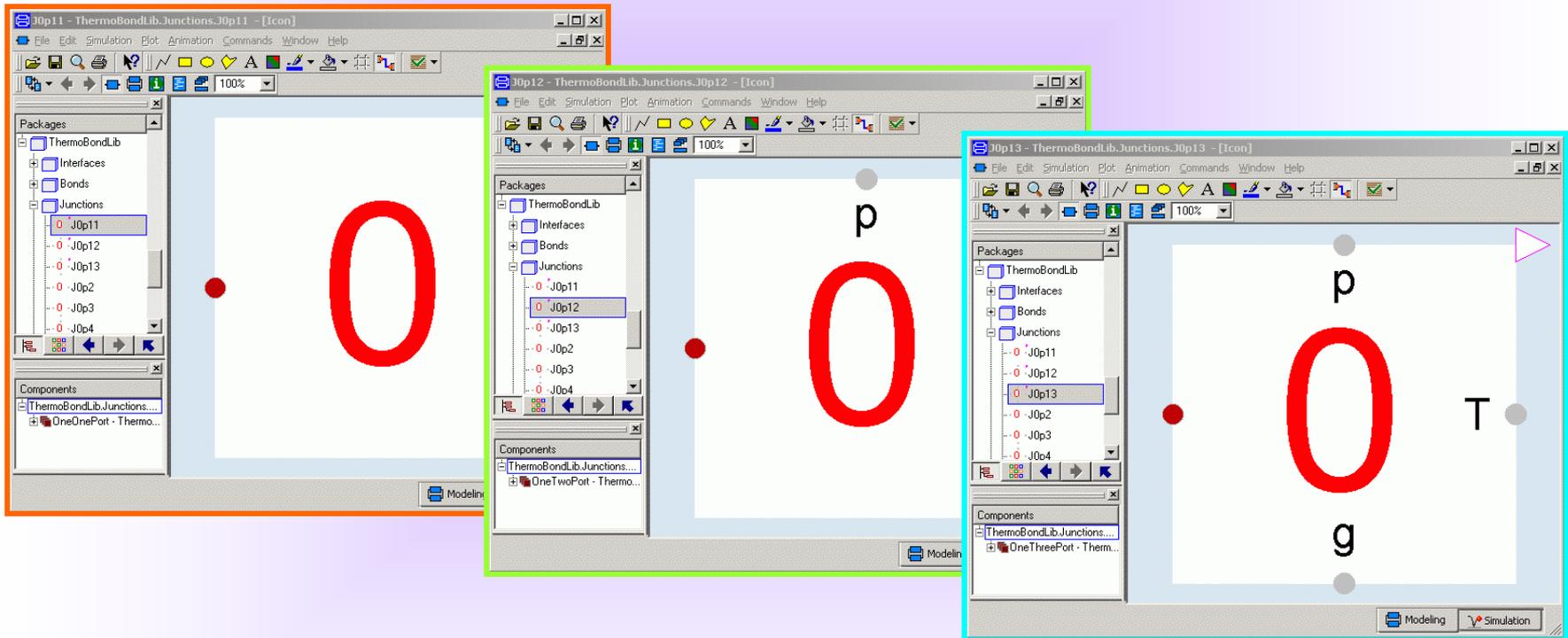
Special Bus-0-Junctions I

```

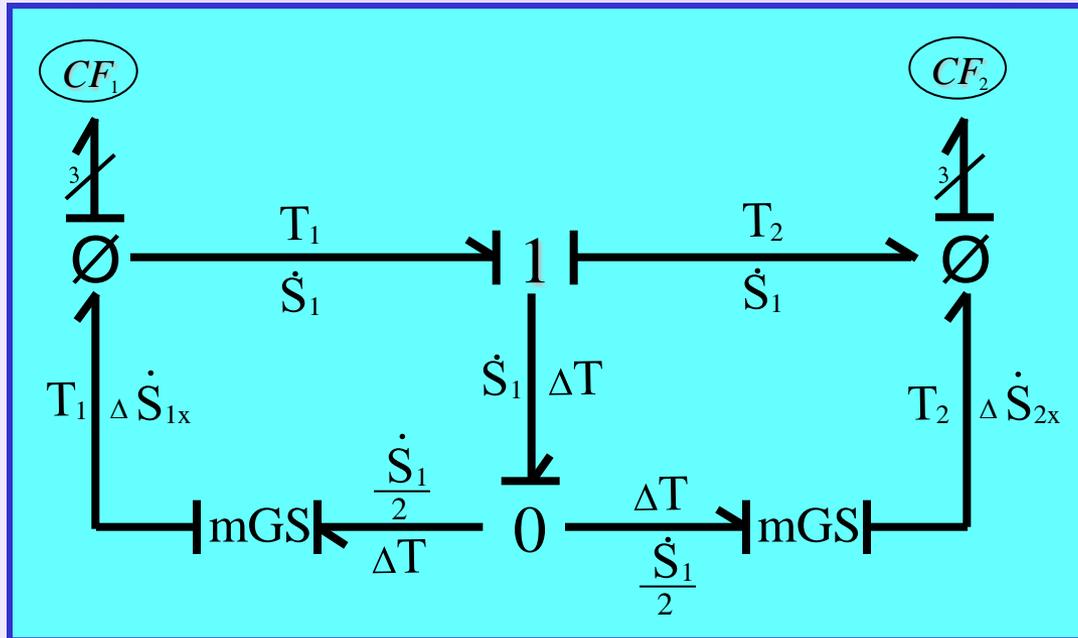
model OneOnePort
  "Partial model invoking one thermo-bond graph connector and one regular bond
  Modelica.SIunits.Temperature T "Temperature";
  Modelica.SIunits.Pressure p "Pressure";
  Modelica.SIunits.SpecificEnthalpy g "Gibbs potential";
  Modelica.SIunits.ThermalConductance Sdot "Entropy flow";
  Modelica.SIunits.VolumeFlowRate q "Volume flow";
  Modelica.SIunits.MassFlowRate Mdot "Mass flow";
  Modelica.SIunits.Entropy S "Entropy";
  Modelica.SIunits.Volume V "Volume";
  Modelica.SIunits.Mass M "Mass";
  Boolean Exist "True if substance exists";
  Real e "Bondgraphic effort";
  Real f "Bondgraphic flow";
equation
  T = ThBondConl.T;
  p = ThBondConl.p;
  g = ThBondConl.g;
  Sdot = ThBondConl.d*ThBondConl.Sdot;
  q = ThBondConl.d*ThBondConl.q;
  Mdot = ThBondConl.d*ThBondConl.Mdot;
  S = ThBondConl.S;
  V = ThBondConl.V;
  M = ThBondConl.M;
  Exist = ThBondConl.Exist;
  BooleanOutPort1 = Exist;
  e = BondConl.e;
  f = BondConl.d*BondConl.f;
end OneOnePort;
  
```



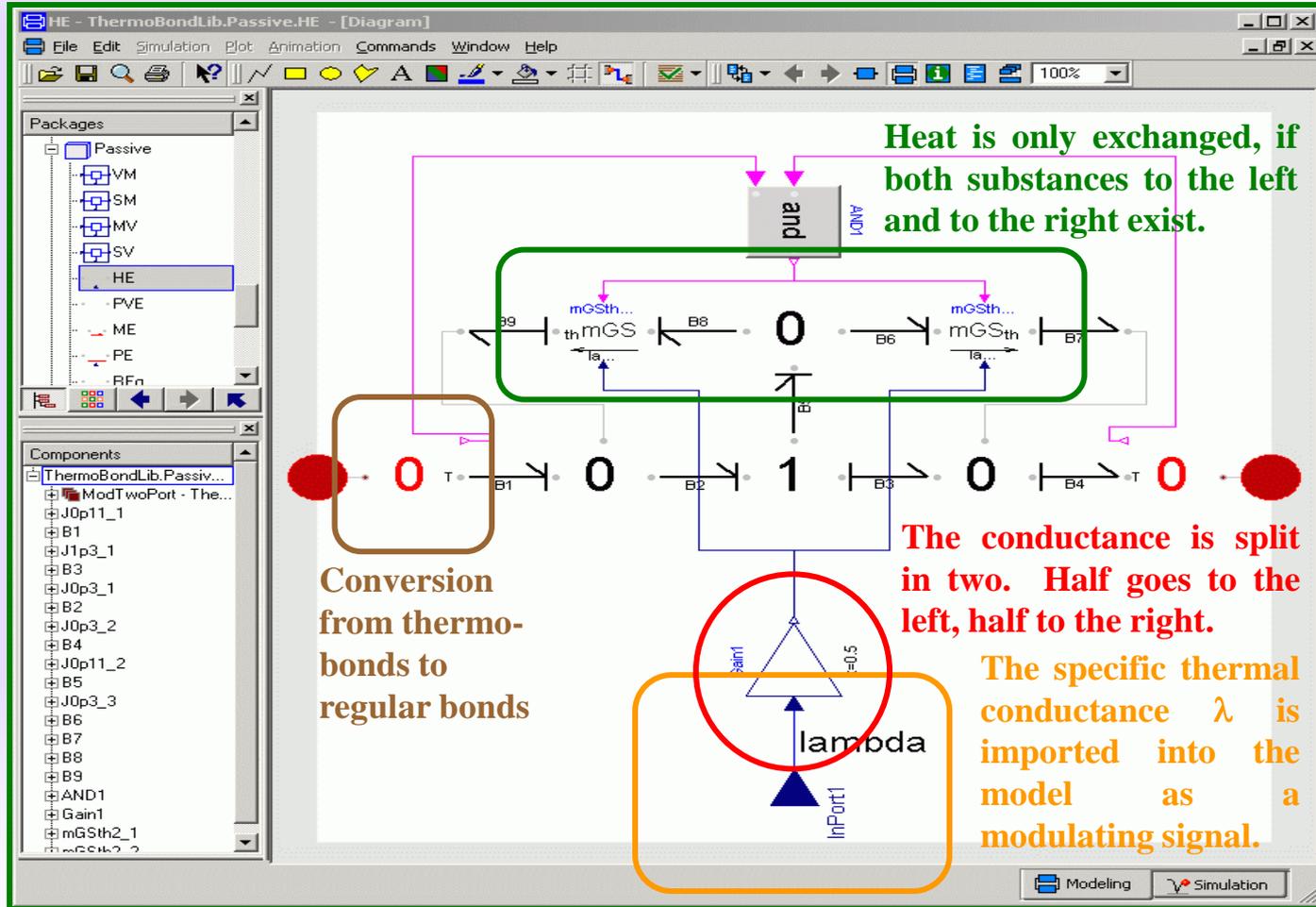
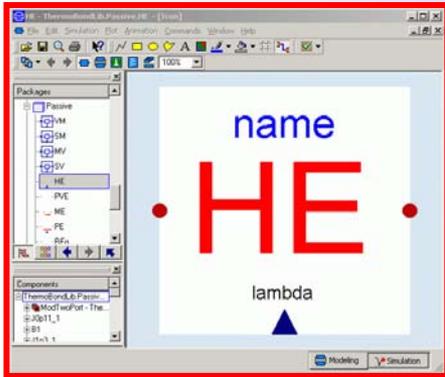
Special Bus-0-Junctions II



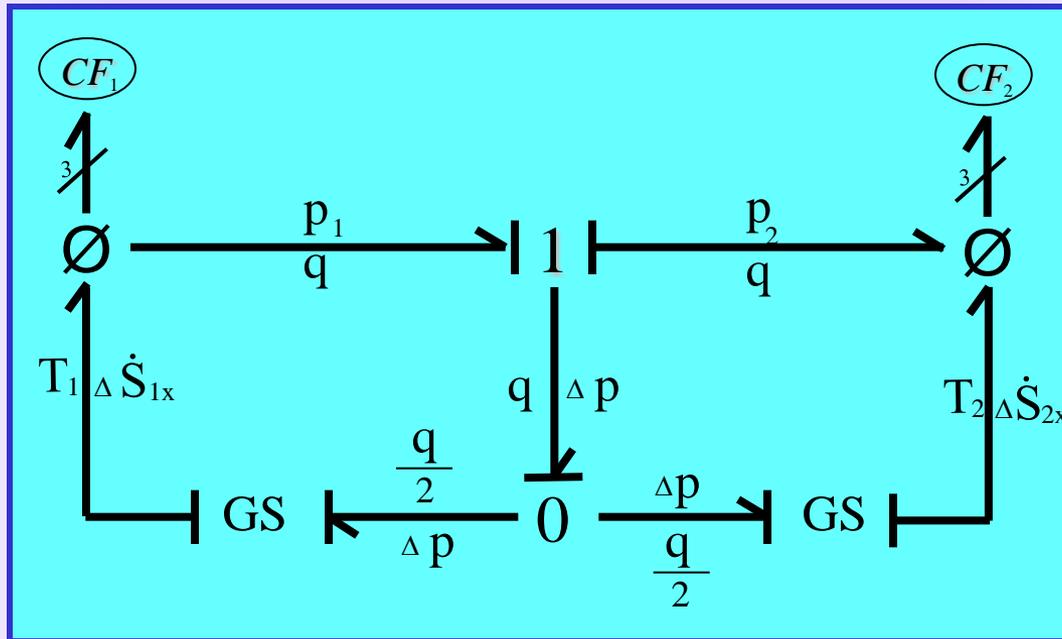
The Heat Exchanger



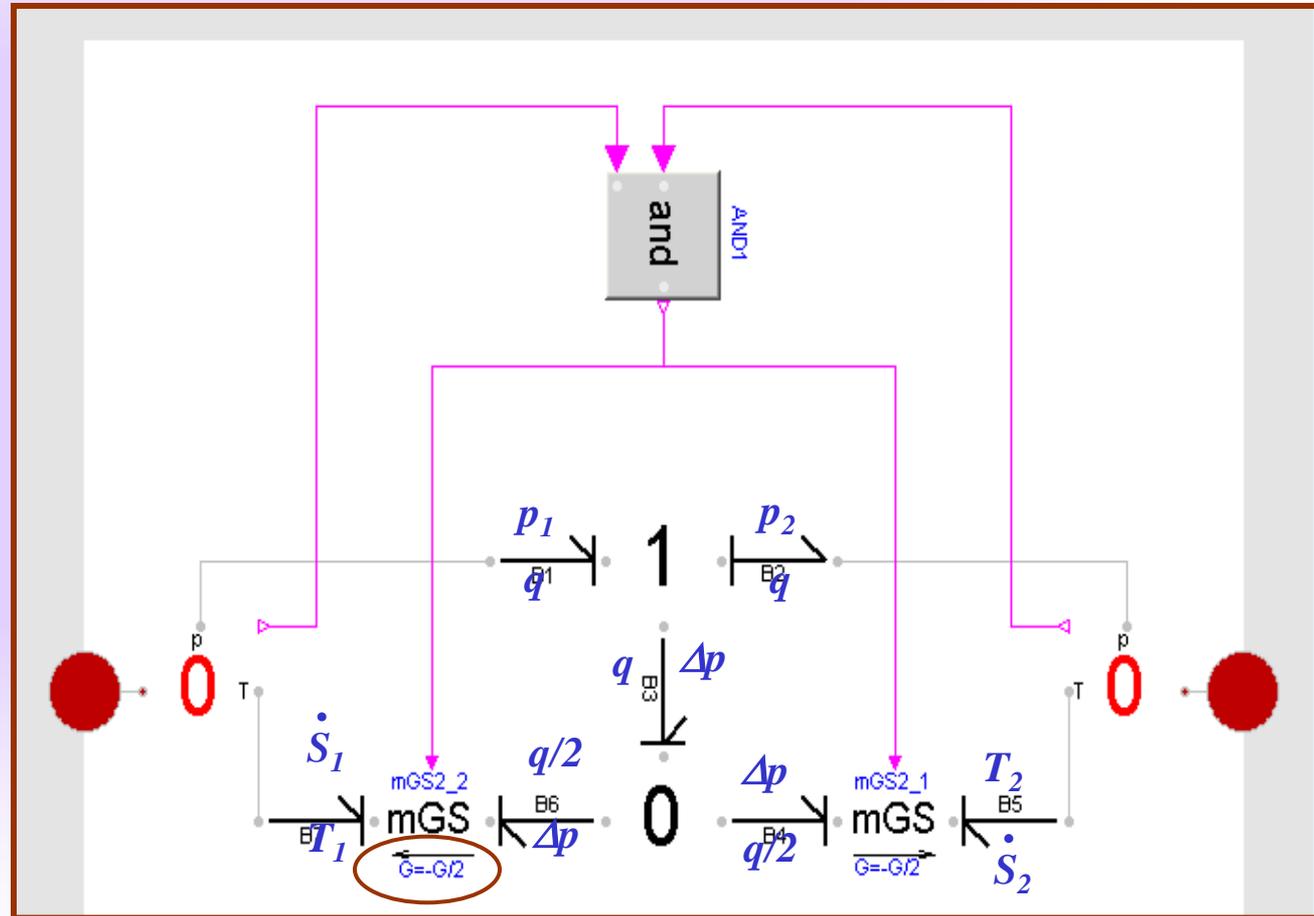
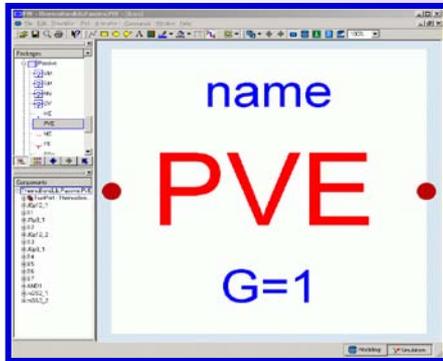
The Thermo-Bond Heat Exchange Element



The Volume Work



The Pressure Volume Exchange Element



We pick up the two pressures.

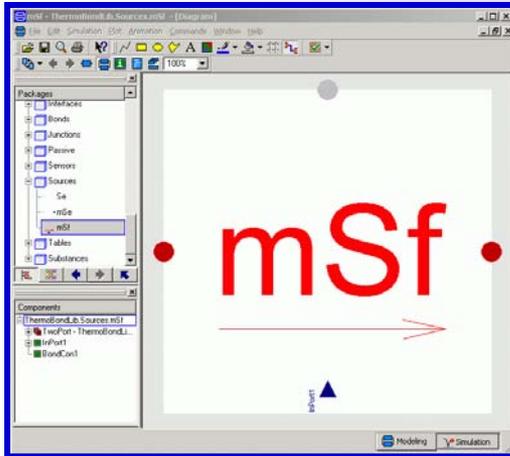
The difference between the two pressures causes a negative volume flow.

The surplus power is split in two to be converted into additional entropy.

The generated entropy is delivered back to the thermal ports.



Forced Volume Flow I



A screenshot of the Modelica Text editor showing the source code for the 'mSf' component. The code is as follows:

```

model mSf "Modulated (simplified) volume flow source"
  extends Interfaces.TwoPort;
  Modelica.SIunits.VolumeFlowRate q "Forced volume flow";
  Boolean sq "True if flow from primary to secondary side";
  Modelica.SIunits.Entropy Sref "Reference Entropy";
  Modelica.SIunits.Volume Vref "Reference Volume";
  Modelica.SIunits.Mass Mref "Reference Mass";
  Modelica.SIunits.Density rho "Density";
  Real s(unit="J/(K.m3)") "Specific entropy";
  Modelica.SIunits.Temperature Tref "Reference temperature";
  Modelica.SIunits.ThermalConductance Sdot "Entropy balance (can be negative)";
  Modelica.SIunits.Pressure Deltap "Pressure difference";
  Boolean Exist "True if there is a flow";
equation
  q = InPort1;
  q1 = q;
  q2 = q1;
  sq = q > 0;
  Sref = if sq then S1 else S2;
  Vref = if sq then V1 else V2;
  Mref = if sq then M1 else M2;
  rho = Mref/Vref;
  Mdot1 = rho*q;
  Mdot2 = Mdot1;
  s = Sref/Vref;
  Sdot1 = s*q;
  Sdot2 = Sdot1;
  Tref = BondConl.e;
  Sdot = -BondConl.d*BondConl.f;
  Tref*Sdot = (T1 - T2)*Sdot1 - (p1 - p2)*q + (g1 - g2)*Mdot1;
  Deltap = p1 - p2;
  Exist = sq and Exist1 or not sq and Exist2;
end mSf;

```

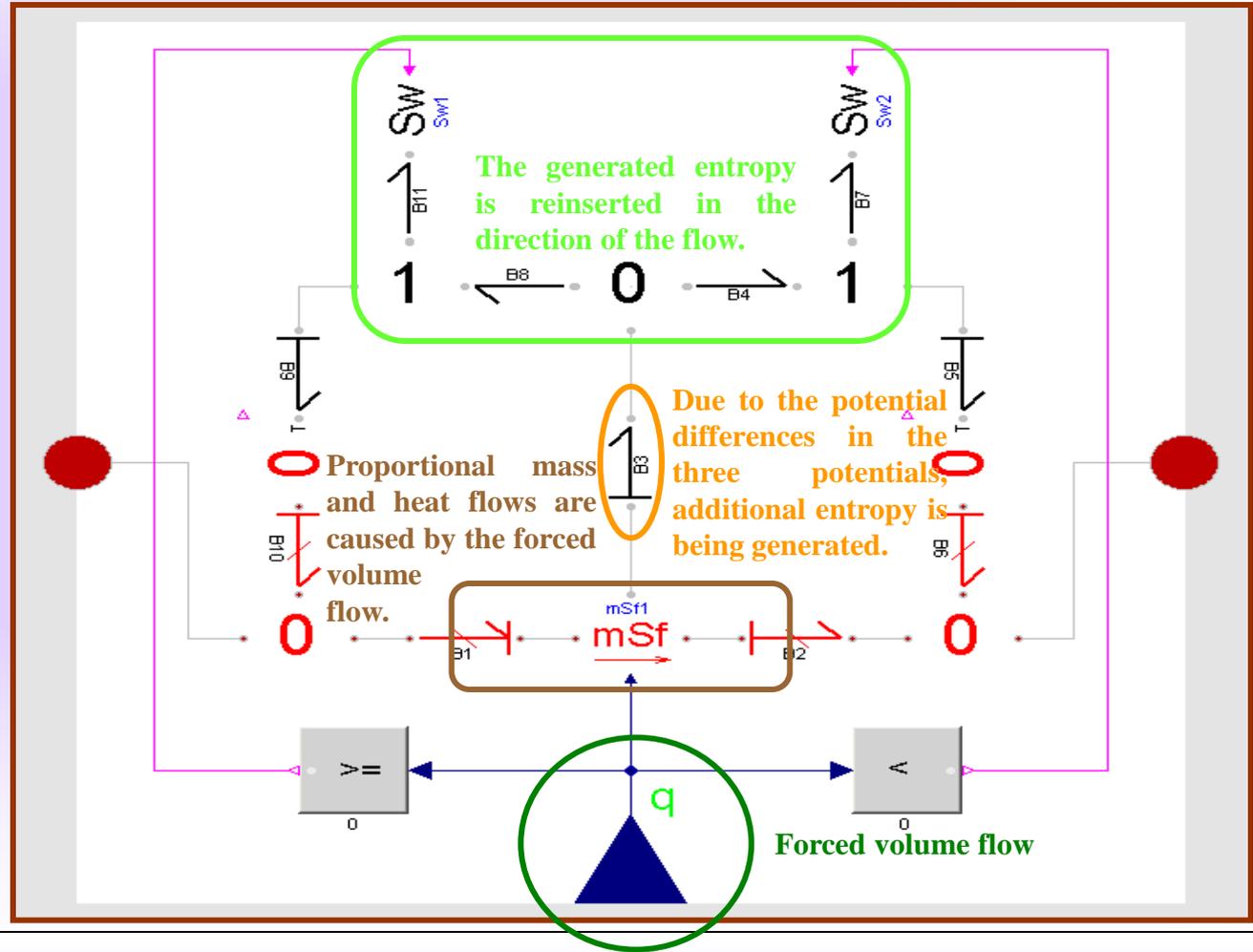
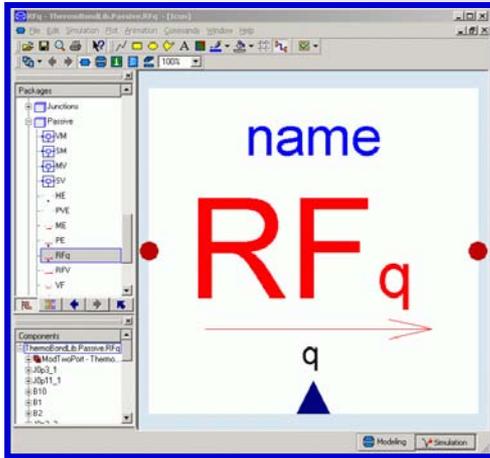
The editor interface includes a package tree on the left, a toolbar at the top, and a status bar at the bottom showing 'Line: 34', 'Modeling', and 'Simulation' modes.

The forced volume flow causes a proportional forced mass flow and an also proportional forced entropy flow.

Forced Volume Flow II

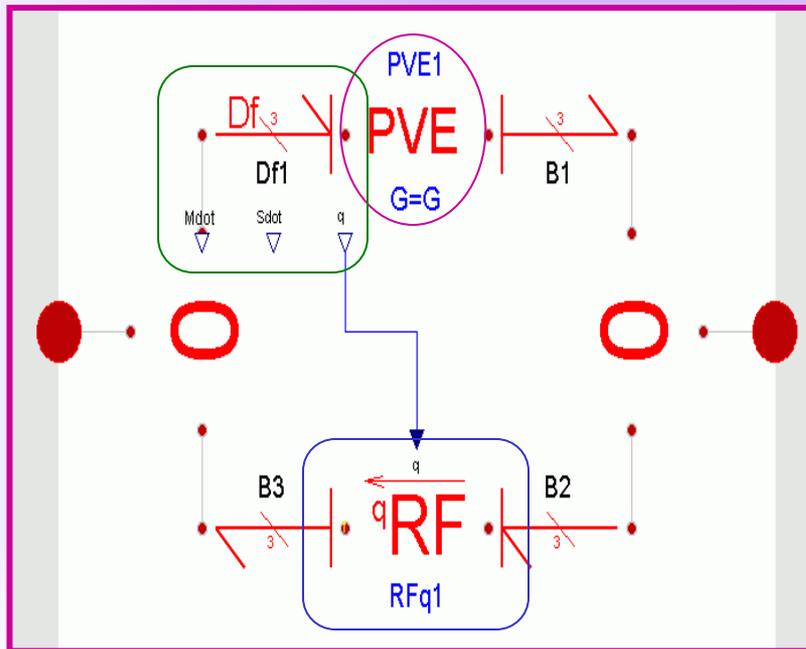
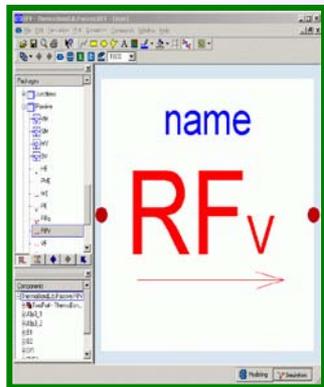
- The model presented here cannot yet be used to represent e.g. a *pump* or a *compressor*, because it doesn't consider the power needed to move the fluid around.
- The model is acceptable to describe small mass movements such as pressure equilibrations between the bulk and a (mathematical) boundary layer.
- An improved forced volume flow model shall be discussed later in this lecture.

The Resistive Field



Pressure Equilibration With Constant Volume

- Sometimes it is useful to allow a mass flow to take place, while the volume doesn't change (remember the gas cartridge).



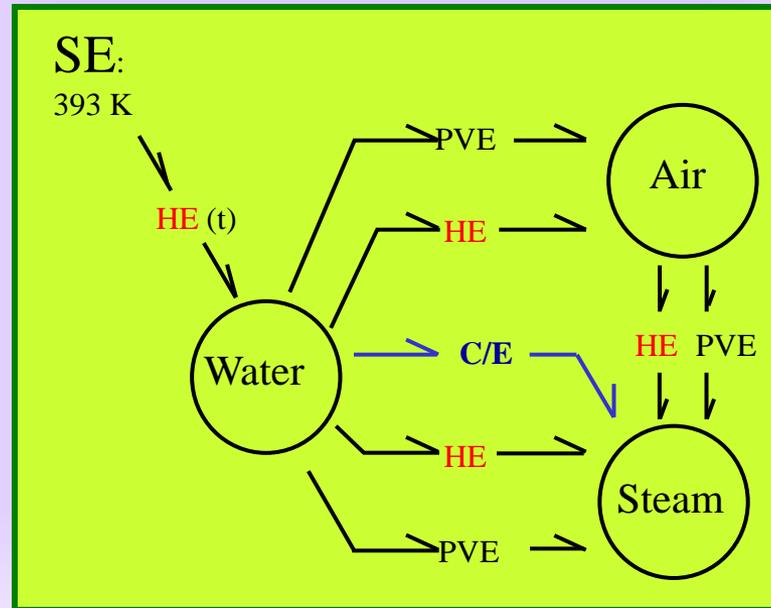
Pressure equilibration causes a volume flow to occur.

A flow sensor element measures the volume flow taking place.

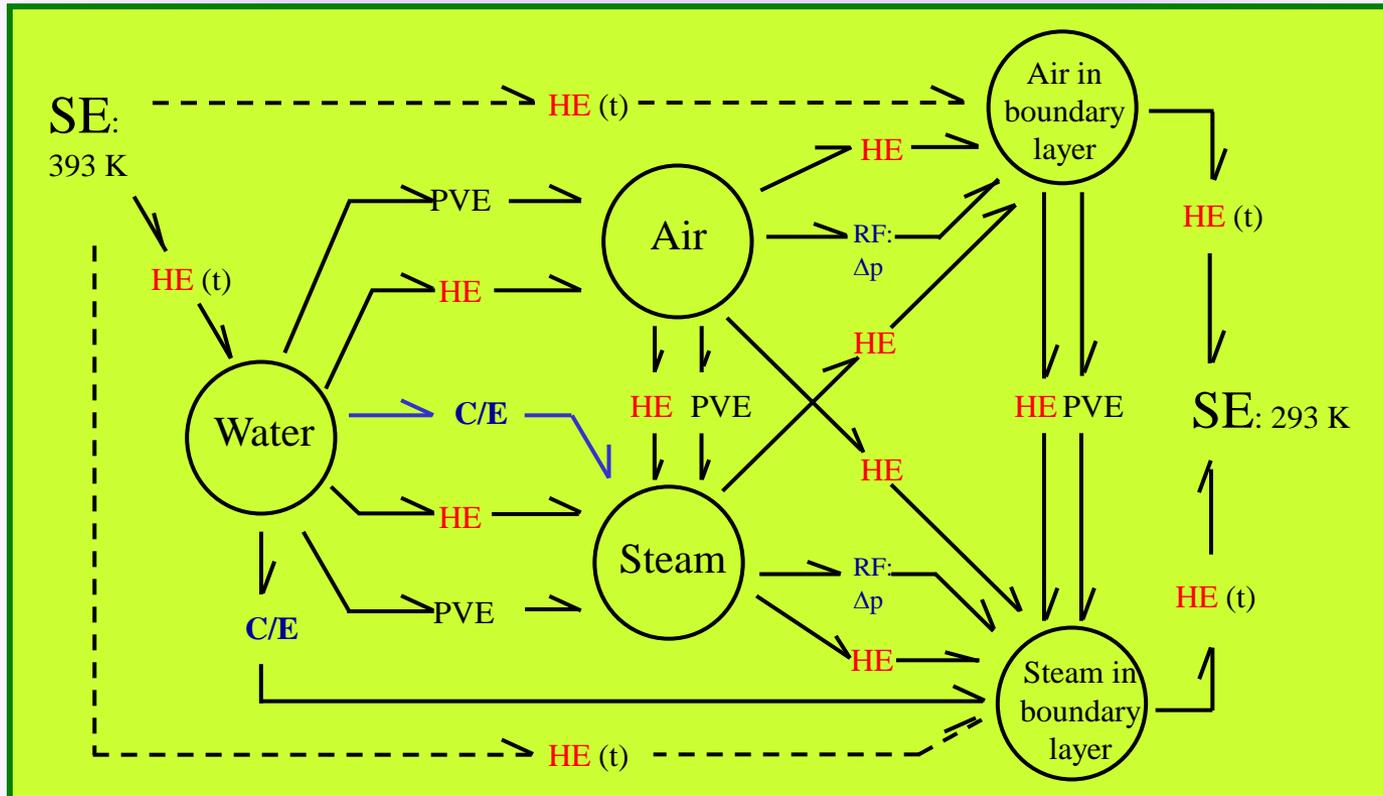
A counter volume flow of equal size is forced.

The pressure gets equilibrated, but the volume remains the same.

The Pressure Cooker I



The Pressure Cooker II



Capacitive Fields

- Let us look at the capacitive field for air.



Linear capacitive field:

$$\underline{\text{der}}(e) = C^{-1} \cdot f$$

By integration:

$$\underline{\text{der}}(q) = f$$

$$\underline{e} = C^{-1} \cdot q$$

Non-linear capacitive field:

$$\underline{\text{der}}(q) = f$$

$$\underline{e} = \underline{e}(q)$$

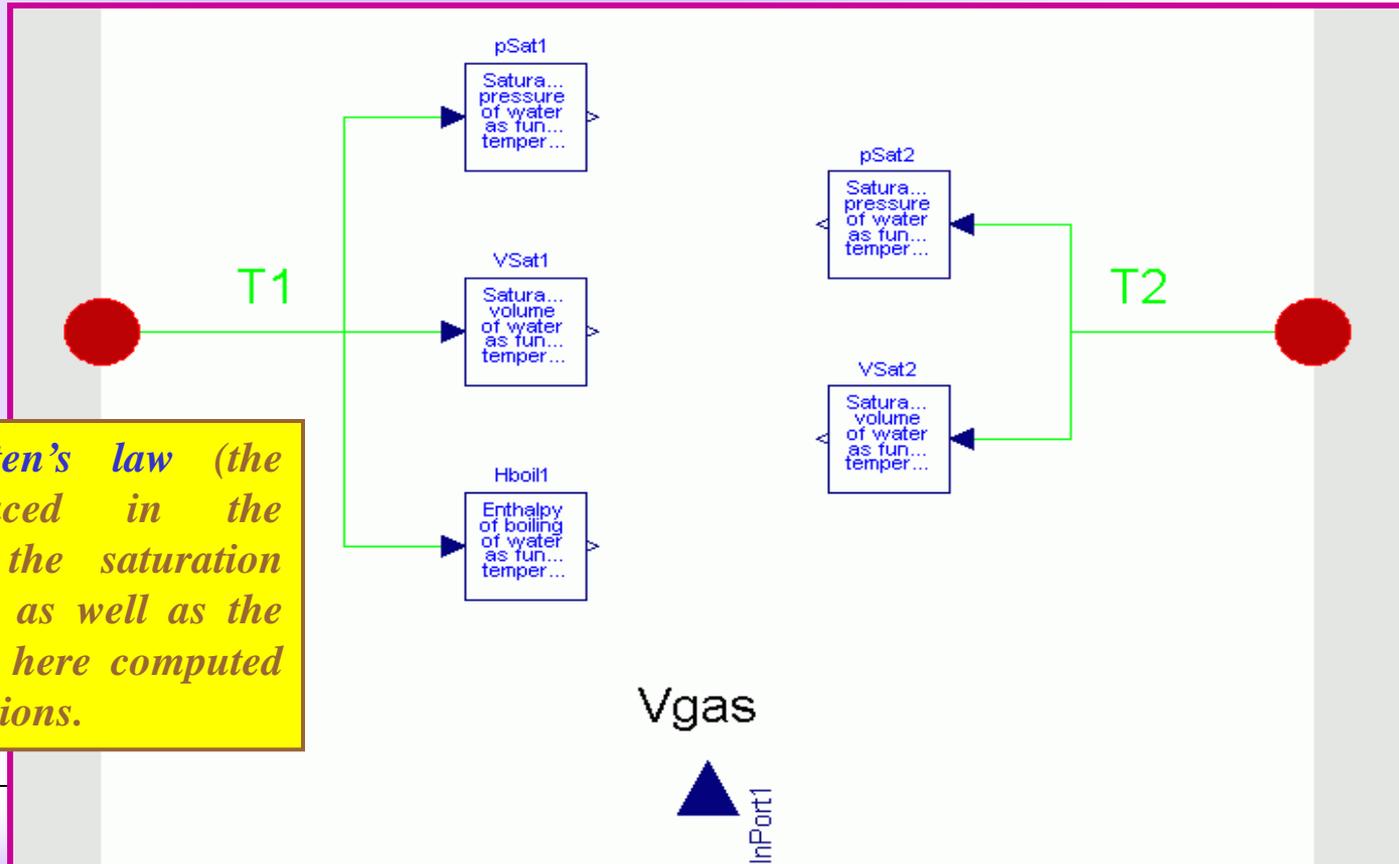
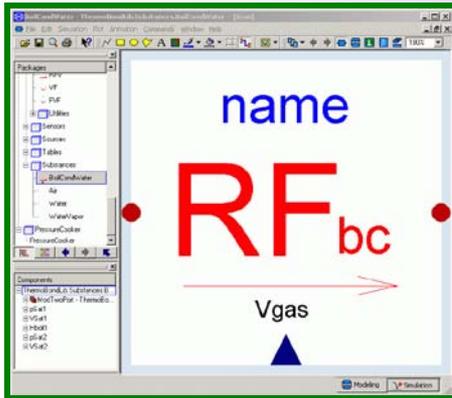
```

model Air "Capacitive field representing air"
  extends Interfaces.PassiveOnePort;
  parameter Modelica.SIunits.Entropy S0=6.81010184 "Entropy if no air";
  parameter Modelica.SIunits.Volume V0=0.83112221e-3 "Volume if no air";
  parameter Modelica.SIunits.Mass M0=1e-3 "Mass if no air";
  parameter Modelica.SIunits.SpecificHeatCapacity cp=1004.0
    "Heat capacity of air at constant pressure";
  parameter Modelica.SIunits.SpecificHeatCapacity R=287.2 "Gas constant";
  parameter Modelica.SIunits.Mass epsM=0.5e-6
    "Smallest mass distinguishable from zero";
  parameter Boolean fict=false "True is fictitious values are used";
  parameter Modelica.SIunits.Temperature T_fict=298.53
    "Fictitious temperature is no air";
  parameter Modelica.SIunits.Pressure p_fict=1e5
    "Fictitious pressure if no air";
  Modelica.SIunits.Entropy S_int "Entropy of air";
  Modelica.SIunits.Volume V_int "Volume of air";
  Modelica.SIunits.Mass M_int "Mass of air";
  Modelica.SIunits.SpecificHeatCapacity cv
    "Heat capacity of air at constant volume";
  Modelica.SIunits.SpecificVolume v "Specific volume";
  Modelica.SIunits.SpecificEntropy s "Specific entropy";
  Real ln_v "Natural logarithm of specific volume";

equation
  der(M_int) = Mdot;
  der(S_int) = Sdot;
  der(V_int) = q;
  Exist = M_int > epsM;
  cv = cp - R;
  v = if Exist then V_int/M_int else 0;
  s = if Exist then S_int/M_int else 0;
  ln_v = Modelica.Math.log(V_int/M_int);
  p = if Exist or not fict then T*R*M_int/V_int else p_fict;
  T = if Exist or not fict then 293.15*exp((s - 6813.7 - R*(ln_v + 0.17245))
    /cv) else T_fict;
  g = T*(cp - s);
  M = if Exist then M_int else M0;
  V = if Exist then V_int else V0;
  S = if Exist then S_int else S0;
end Air;
  
```

Evaporation and Condensation I

- The models describing *evaporation* and *condensation* are constructed by interpolation from steam tables.



Instead of using Teten's law (the approximation embraced in the Biosphere 2 model), the saturation pressures and volumes, as well as the enthalpy of boiling are here computed using table-lookup functions.

Evaporation and Condensation II

```

model BoilCondWater "Boiling and condensation of water"
  extends Interfaces.ModTwoPort(InPort, OutPort,
    Modelica.SIunits.Volume);
  parameter Real Rb(unit="m.s") = 0.01;
  parameter Real Rc(unit="m.s") = 0.01;
  Modelica.SIunits.Pressure pSat_liq;
  Modelica.SIunits.Pressure pSat_gas;
  Modelica.SIunits.SpecificVolume VSat_liq;
  Modelica.SIunits.SpecificVolume VSat_gas;
  Modelica.SIunits.SpecificEnthalpy hboil;
  Modelica.SIunits.Volume Vgas "Gas volume";
  Modelica.SIunits.Pressure pgas "Partial pressure";
  Modelica.SIunits.MassFlowRate Mboil;
  Modelica.SIunits.MassFlowRate Mcond;
  Boolean fd "True if forward flow";
  Modelica.SIunits.HeatFlowRate DeltaQdot "Heat generated by potential equilibrium";
  Boolean Exist "True if flow exists";
  Modelica.SIunits.Mass Mref "Reference mass";
  Modelica.SIunits.AngularFrequency xi;
  Modelica.SIunits.ThermalConductance k;
  Modelica.SIunits.ThermalConductance k;
  Modelica.SIunits.Volume Vsteam "Maximum steam volume";
  Real hum1 "Relative humidity computed from pgas";
  Real hum2 "Relative humidity computed from Vgas";

equation
  pSat1.u = T1;
  pSat2.u = T2;
  VSat1.u = T1;
  VSat2.u = T2;
  Hboill.u = T1;
  pSat_liq = pSat1.y;
  pSat_gas = pSat2.y;
  VSat_liq = VSat1.y;
  VSat_gas = VSat2.y;
  hboil = Hboill.y;
  Vgas = s;
  pgas = p2*V2/Vgas;
  Mboil = if ((pSat_liq > p1) and Exist1) then Rb*(pSat_liq - p1) else 0;
  Mcond = if ((pSat_gas < pgas) and Exist2) then Rc*(pgas - pSat_gas) else 0;
  Mdot1 = Mboil - Mcond;
  Mdot2 = Mdot1;
  q1 = Mboil*VSat_liq - Mcond*VSat_gas;
  q2 = q1;
  fd = Mdot1 > 0;
  DeltaQdot = (q1 - q2)*Mdot1 - (p1 - p2)*q1;
  Exist = (fd and Exist1) or (not fd and Exist2);
  Mref = if fd then M1 else M2;
  xi = if Exist then Mdot1/Mref else 0;
  Sdot2_aux = if Exist2 then Sdot1 + (DeltaQdot + (T1 - T2)*Sdot1)/T1 else Sdot1;
  Sdot1_aux = if Exist1 then Sdot2 + (DeltaQdot + (T1 - T2)*Sdot2)/T2 else Sdot2;
  Sdot1 = if fd then xi*S1 + hboil*Mdot1/T1 else Sdot1_aux;
  Sdot2 = if fd then Sdot2_aux else xi*S2 - hboil*Mdot2/T2;
  Vsteam = 1/(p2/pSat_gas - 1)*Vgas;
  hum1 = pgas/pSat_gas;
  hum2 = 1 - V2/Vsteam;
end BoilCondWater;

```

A bit messy!

Simulation of Pressure Cooker

- We are now ready to compile and simulate the model.

Messages - Dymola

Syntax Error Translation Dialog Error Simulation

Translation of ThermoBondLib.Example.PressureCooker:
DAE having 10556 scalar unknowns and 10556 scalar equations.

STATISTICS

Original Model
Number of components: 625
Variables: 9605
Constants: 0
Parameters: 179 (733 scalars)
Unknowns: 9426 (10556 scalars)
Differentiated variables: 15 scalars
Equations: 6987
Nontrivial: 3745

Translated Model
Constants: 2248 scalars
Free parameters: 714 scalars
Parameter depending: 31 scalars
Inputs: 0
Outputs: 14 scalars
Continuous time states: 15 scalars
Time-varying variables: 447 scalars
Alias variables: 7851 scalars
Assumed default initial conditions: 15
LogDefaultInitialConditions=true; gives more information
Number of mixed real/discrete systems of equations: 0
Sizes of linear systems of equations: {4, 4}
Sizes after manipulation of the linear systems: {0, 0}
Sizes of nonlinear systems of equations: {6, 6, 6, 6}
Sizes after manipulation of the nonlinear systems: {3, 3, 3, 3}
Number of numerical Jacobians: 0

Experiment Setup

General Translation Output Debug Compiler Realtime

Experiment
Name: PressureCooker

Simulation interval
Start time: 0
Stop time: 20000

Output interval
 Interval length: 0
 Number of intervals: 500

Integration
Algorithm: Dassl
Tolerance: 1e-010
Fixed Integrator Step: 0

OK Store in model Cancel

Messages - Dymola

Syntax Error Translation Dialog Error Simulation

Log-file of program ./dymosim
(generated: Thu Feb 07 23:33:49 2008)

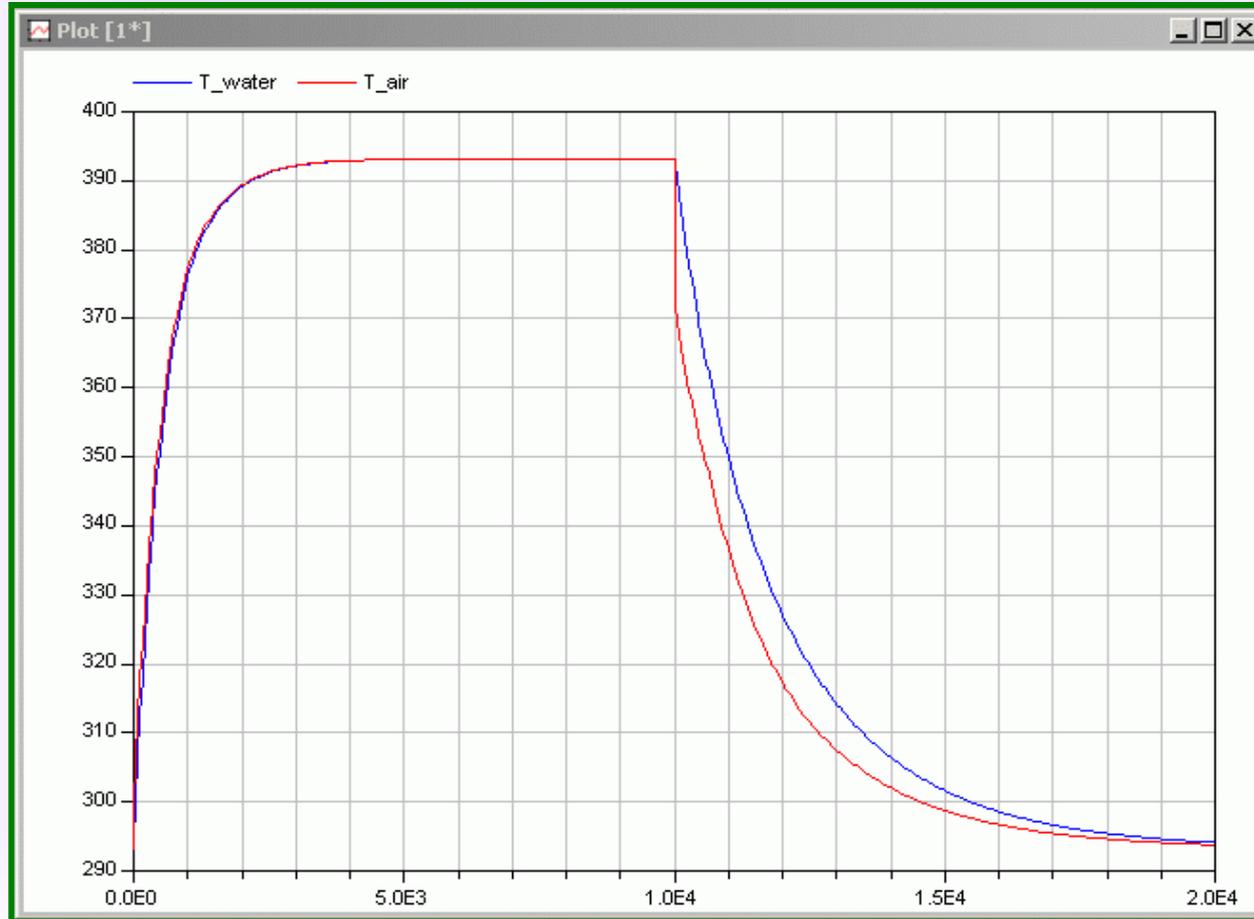
dymosim started
... "dsin.txt" loading (dymosim input file)
... "PressureCooker.mat" creating (simulation result file)

Integration started at T = 0 using integration method DASSL
(DAE multi-step solver (dassl/dasslrt of Petzold modified by Dymasim))
Integration terminated successfully at T = 20000

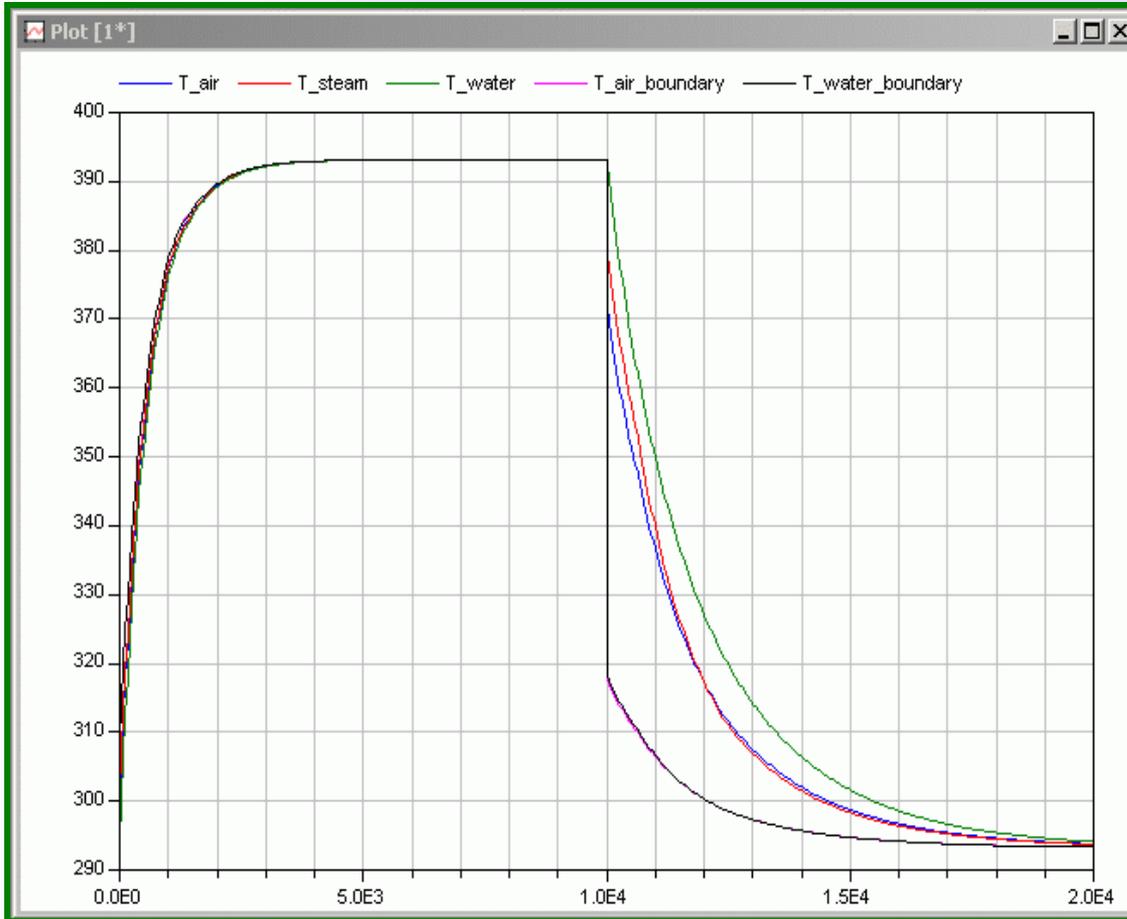
CPU-time for integration: 1.54 seconds
CPU-time for one GRID interval: 3.06 mill-seconds
Number of result points: 522
Number of GRID points: 501
Number of (successful) steps: 2814
Number of F-evaluations: 15458
Number of H-evaluations: 3376
Number of Jacobian-evaluations: 625
Number of (model) time events: 1
Number of (U) time events: 0
Number of state events: 10
Number of step events: 0
Minimum integration stepsize: 1.9e-009
Maximum integration stepsize: 100
Maximum integration order: 5

Calling terminal section
... "dsfinal.txt" creating (final states)

Simulation Results



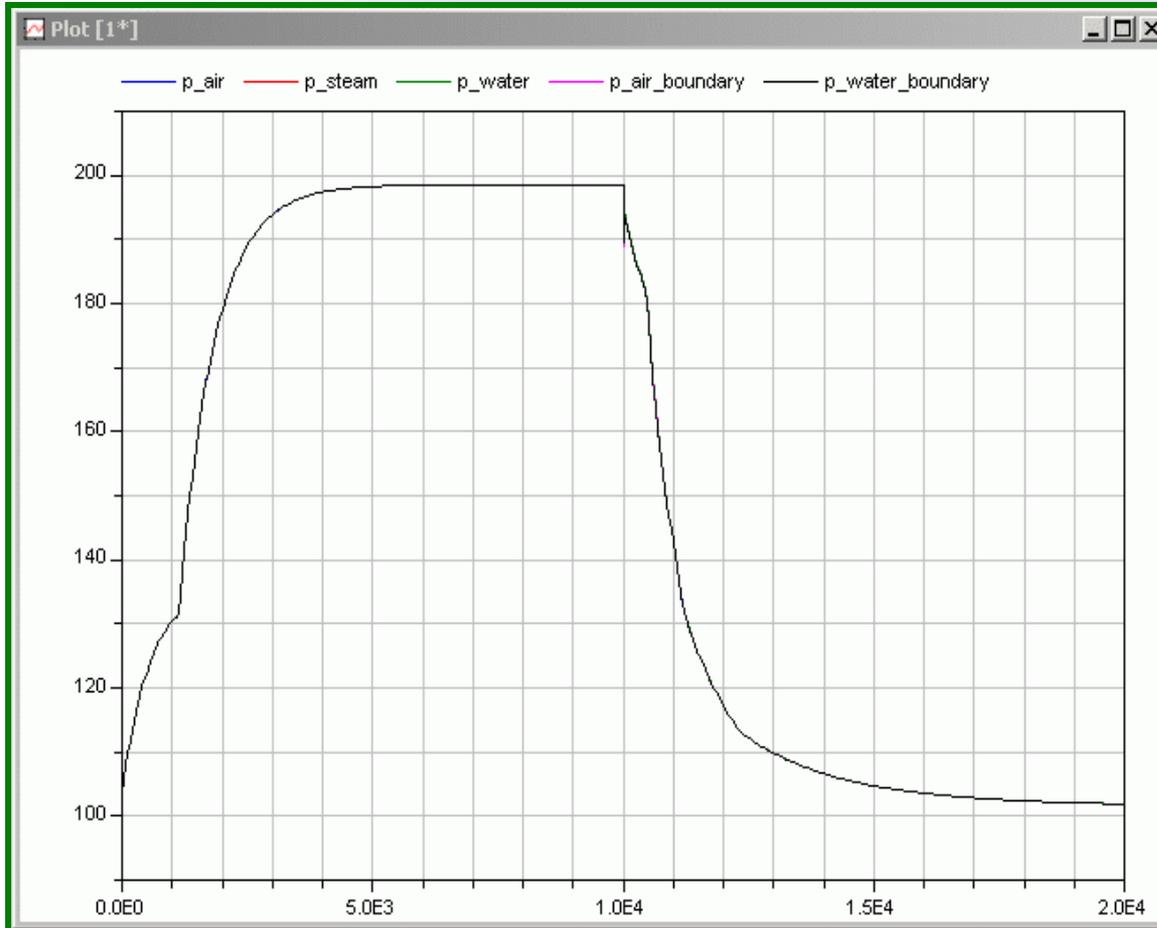
Simulation Results II



Heating is sufficiently slow that the temperature values of the different media are essentially indistinguishable. The heat exchangers have a smaller time constant than the heating.

During the cooling phase, the picture is very different. When cold water is poured over the pressure cooker, air and steam in the small boundary layer cool down almost instantly. Air and steam in the bulk cool down more slowly, and the liquid water cools down last.

Simulation Results III

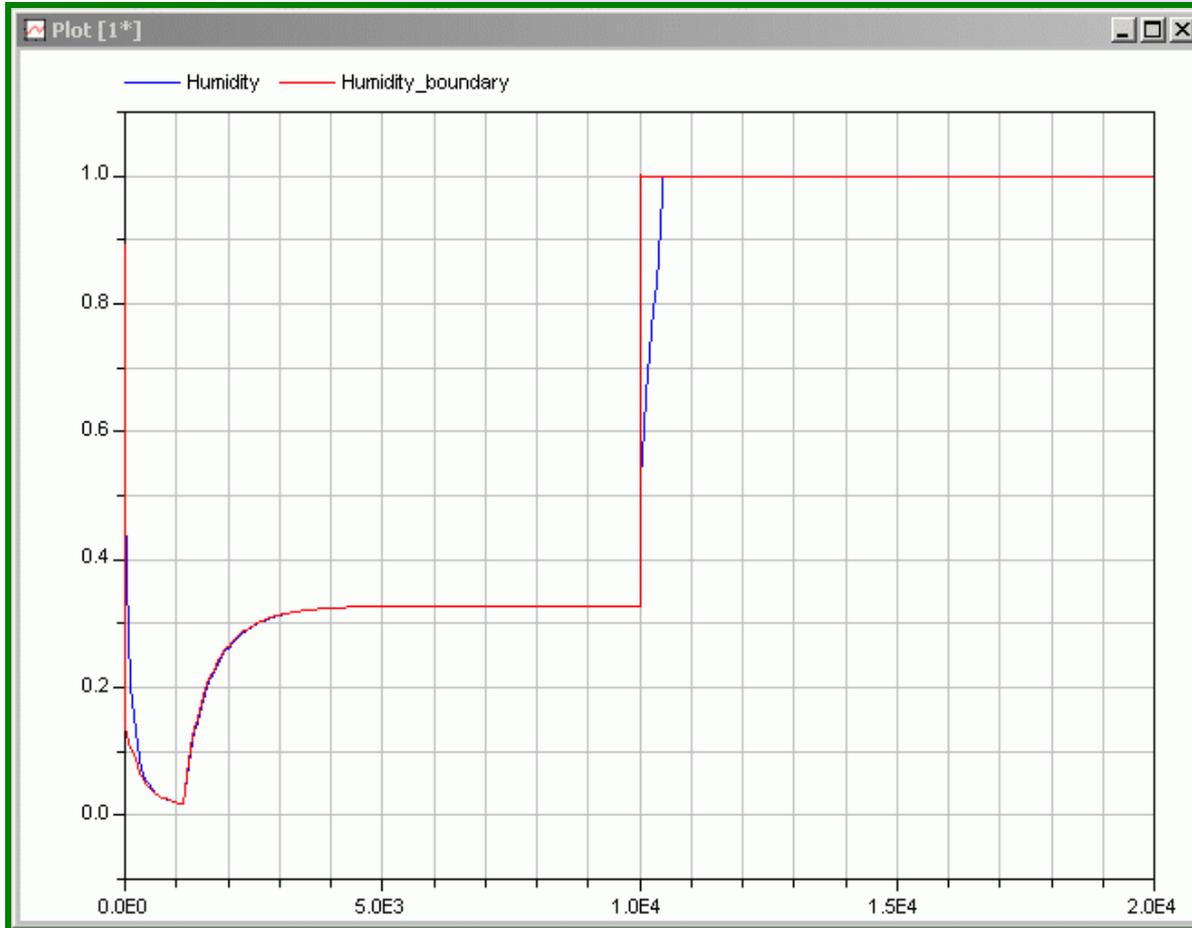


The pressure values are essentially indistinguishable throughout the simulation.

During the heating phase, the pressures rise first due to rising temperature. After about 150 seconds, the liquid water begins to boil, after which the pressure rises faster, because more steam is produced (water vapor occupies more space at the same temperature than liquid water).

The difference between boundary layer and bulk pressure values in the cooling phase is a numerical artifact.

Simulation Results IV

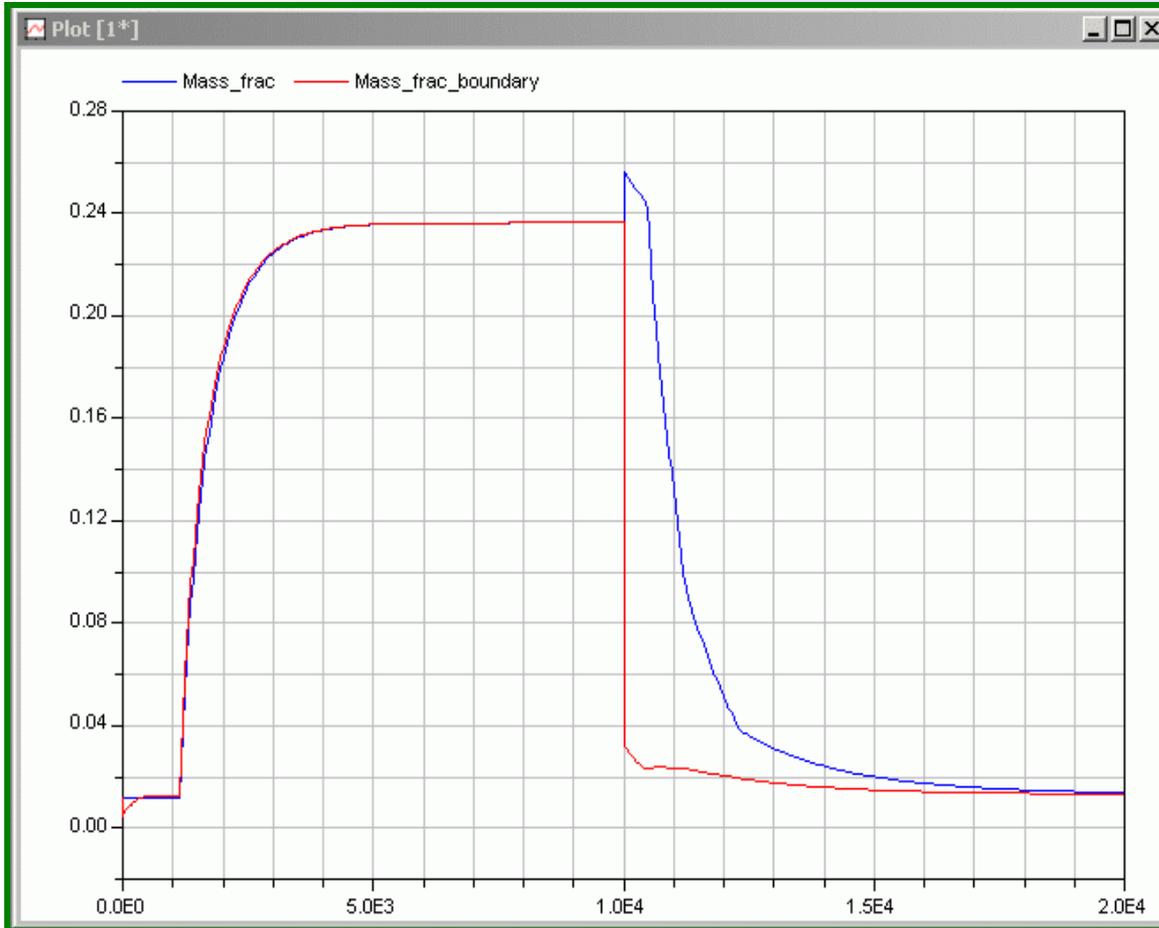


The relative humidity decreases at first, because the saturation pressure rises with temperature, i.e., more humidity can be stored at higher temperatures.

As boiling begins, the humidity rises sharply, since additional vapor is produced.

In the cooling phase, the humidity quickly goes into saturation, and stays there, because the only way to ever get out of saturation again would be by reheating the water.

Simulation Results V



The *mass fraction* defines the percentage of water vapor contained in the air/steam mixture.

Until the water begins to boil, the mass fraction is constant. It then rises rapidly until it reaches a new equilibrium, where evaporation and condensation balance out.

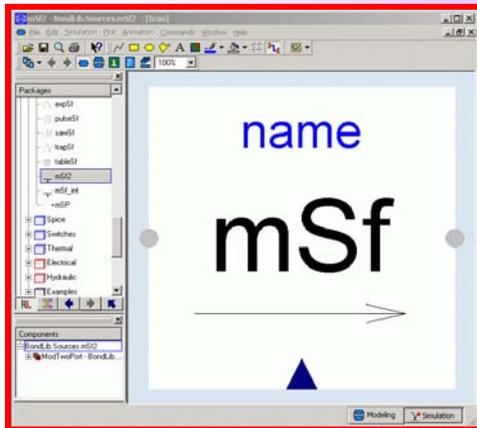
During the cooling phase, the boundary layer cools down quickly, and can no longer hold the water vapor contained. Some falls out as water, whereas other steam gets pushed into the bulk, temporarily increasing the mass fraction there even further.

Free Convective Mass Flow

- We are now ready to discuss *free convective mass flow*, such as mass flow occurring in a segment of a pipe.
- The convective mass flow occurs because more mass is pushed in from one end, pushing the mass currently inside the pipe segment out by the other end.
- To this end, we need to introduce some more models.

The Forced Flow Source

- This model describes an element of the regular bond-graph library.



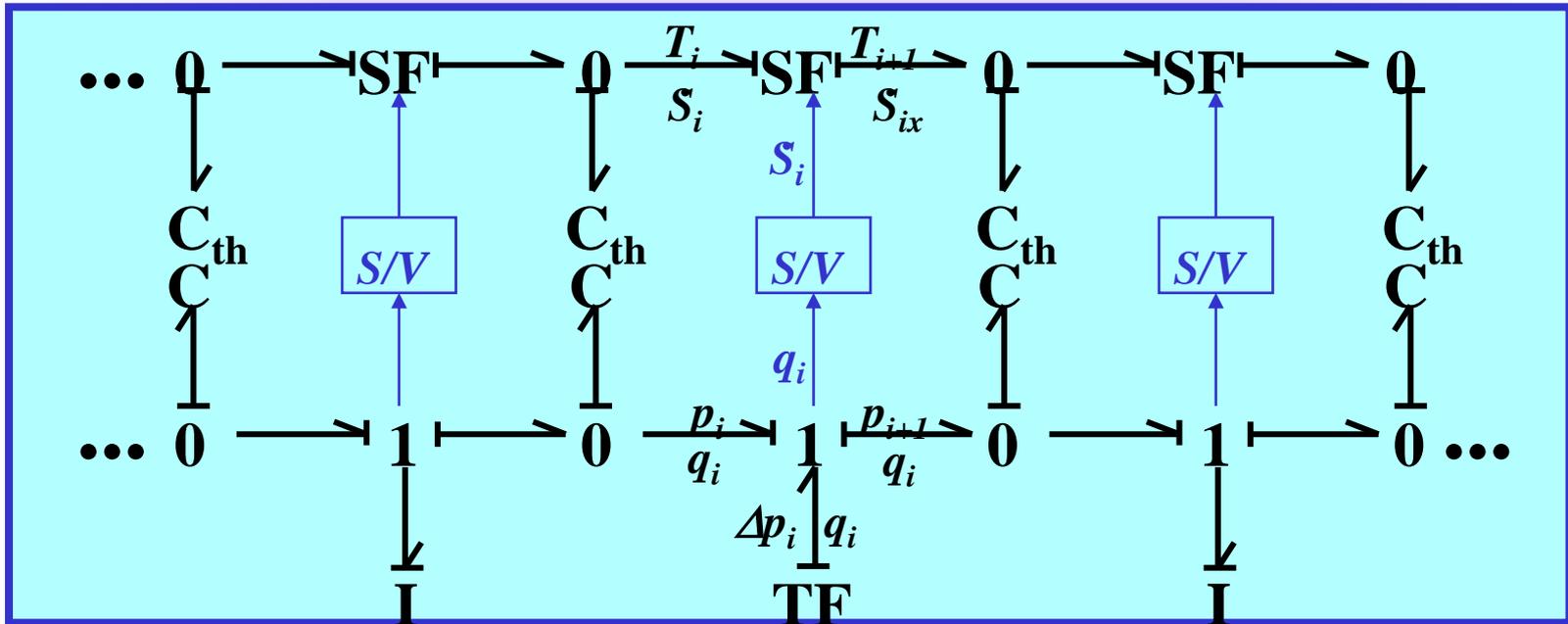
The primary side is a flow source, the secondary side can be either a flow or an effort source. Its equation is defined to satisfy power continuity across the element.

A screenshot of the Modelica Text Editor showing the 'mSf2' model code. The code is as follows:

```
model mSf2
  extends Interfaces.ModTwoPort;
equation
  f1 = s;
  e1*f1 = e2*f2;
end mSf2;
```

The screenshot also shows the package tree on the left, the toolbar, and the status bar at the bottom.

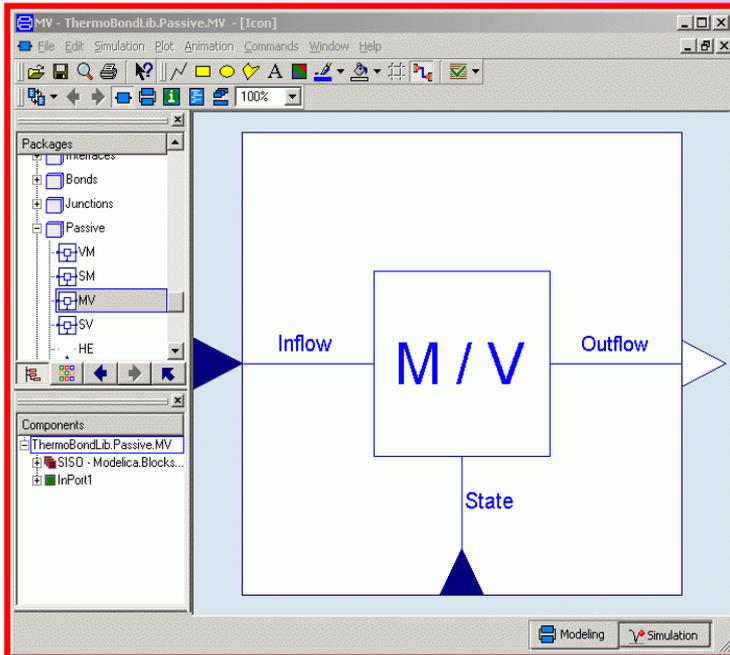
Density and Specific Entropy I



- As mentioned some lectures ago, we shall need modulated flow sources (as introduced one slide ago) that are modulated by the specific entropy and/or the specific mass (i.e., the density).

Density and Specific Entropy II

- These models are created as blocks:



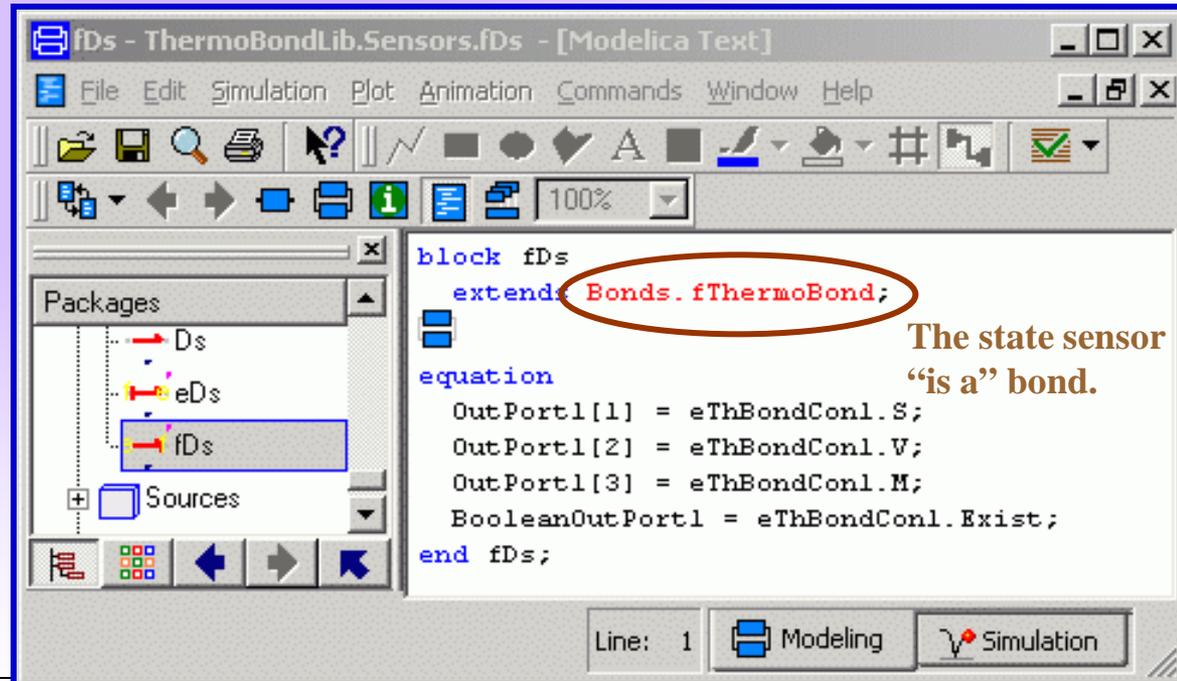
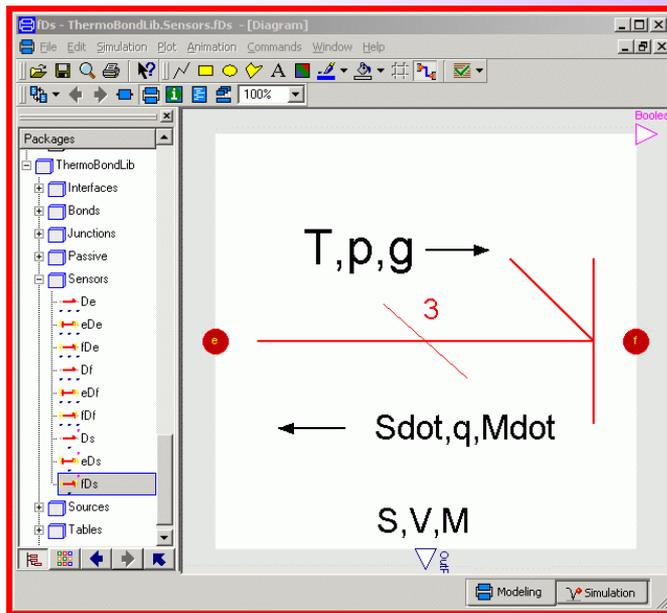
```

block MV "Volume flow to mass flow converter"
  extends Modelica.Blocks.Interfaces.SISO(y(redeclare type SignalType =
    Modelica.SIunits.MassFlowRate), u(redeclare type SignalType =
    Modelica.SIunits.VolumeFlowRate));
  parameter Modelica.SIunits.VolumeFlowRate eps_v = Modelica.Constants.small
    "Smallest absolute volume flow considered unequal zero";
  output Modelica.SIunits.Entropy S "Entropy";
  output Modelica.SIunits.Volume V "Volume";
  output Modelica.SIunits.Mass M "Mass";
  output Modelica.SIunits.VolumeFlowRate f1 "Volume inflow";
  output Modelica.SIunits.MassFlowRate f2 "Mass outflow";
  output Boolean Exist;
equation
  S = InPort1[1];
  V = InPort1[2];
  M = InPort1[3];
  Exist = abs(u) > eps_v;
  f1 = u;
  f2 = if Exist then (M/V)*f1 else 0;
  y = f2;
end MV;

```

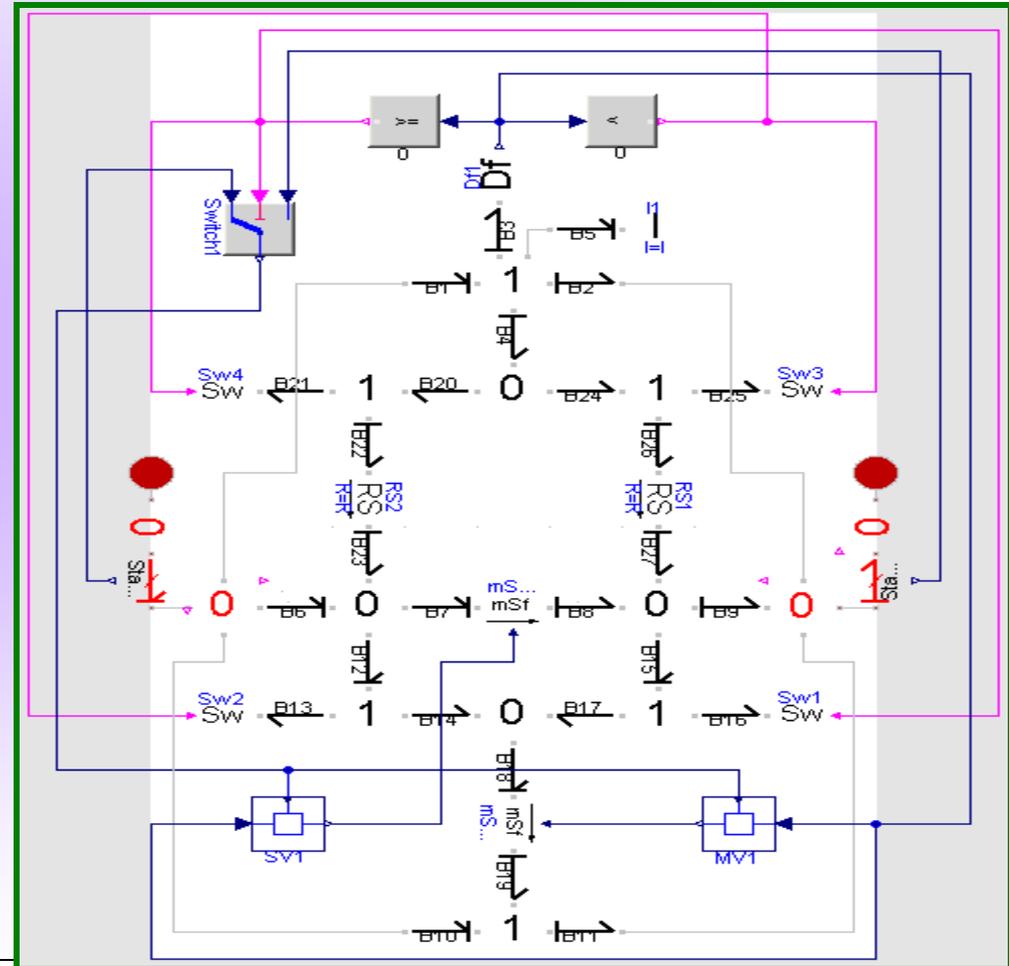
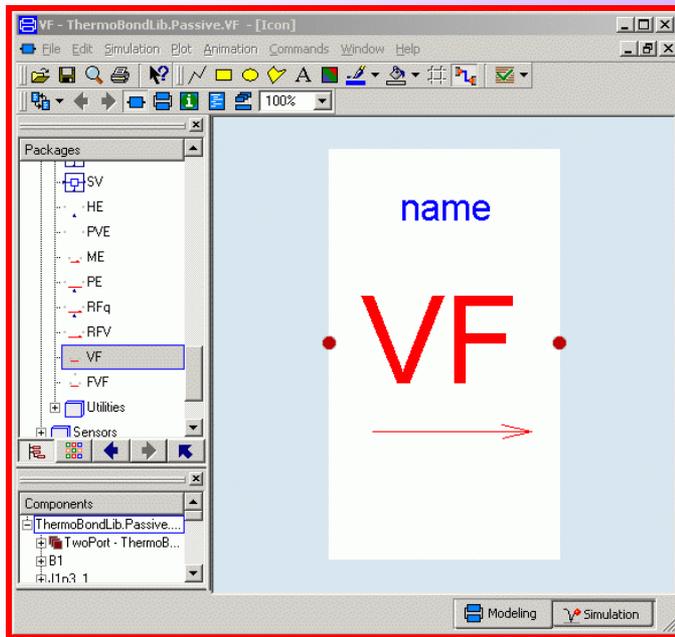
The State Sensor

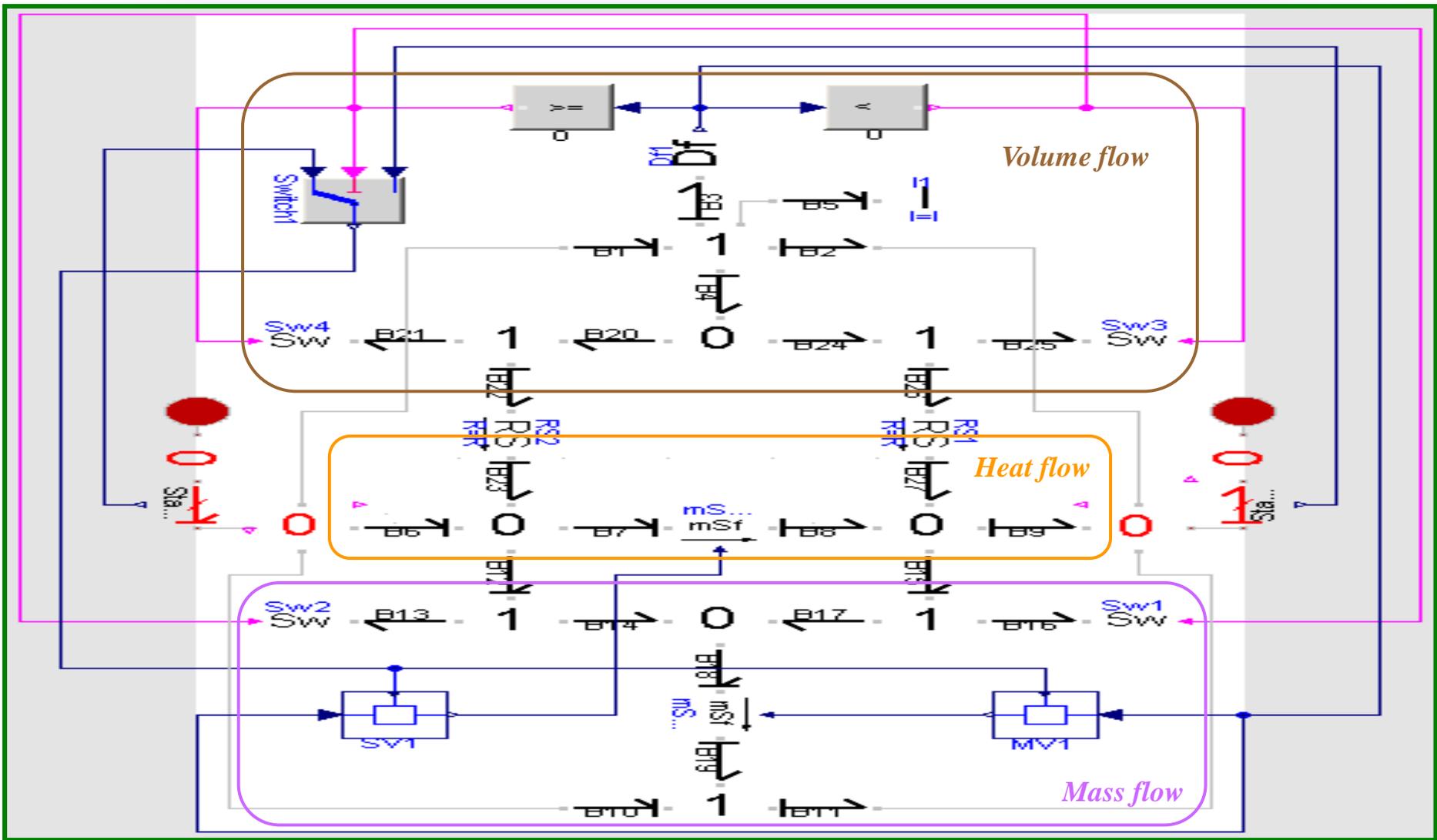
- Many elements that are related to substances require state information. This is generated by a specialized thermo-bond, the so-called *state sensor element*.

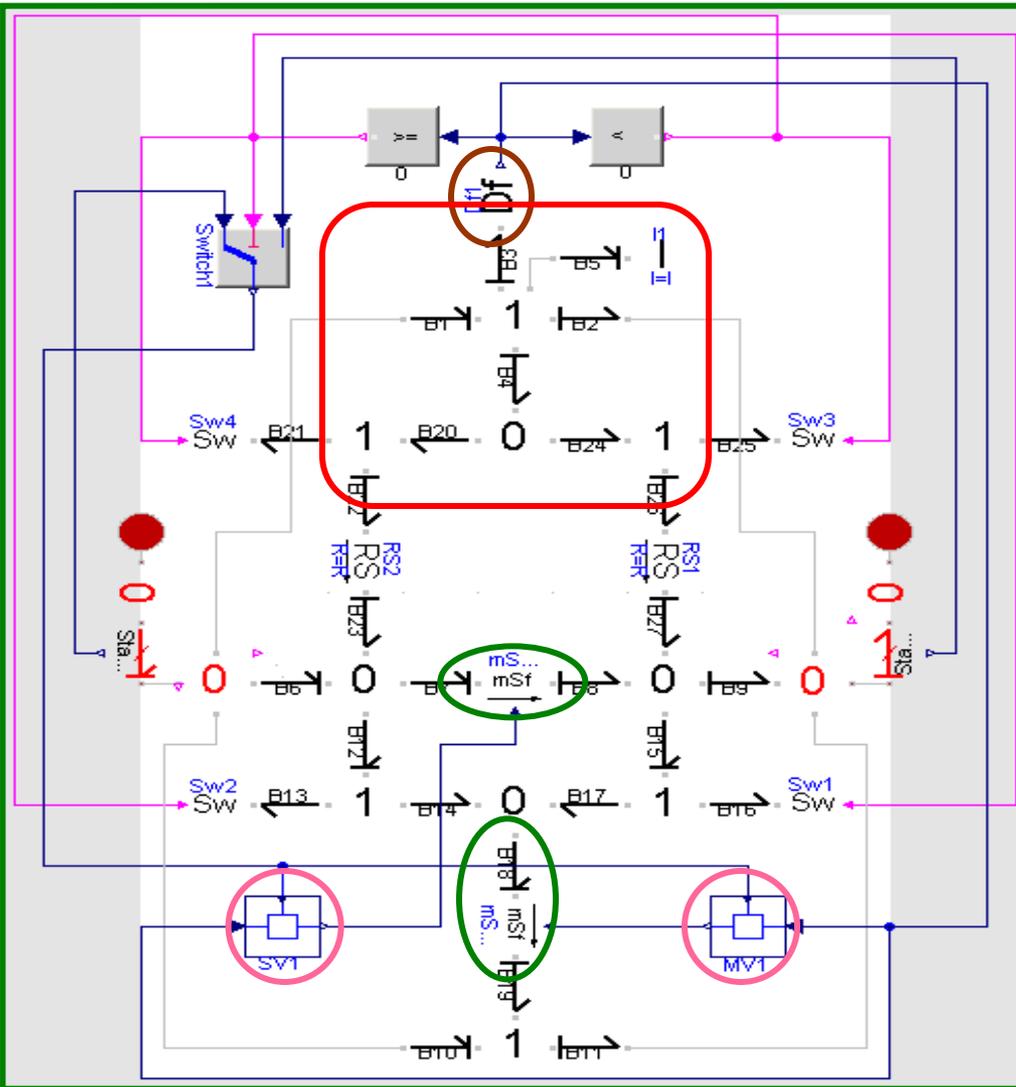


Free Convective Volume Flow

- We are now ready to describe the free convective volume flow.





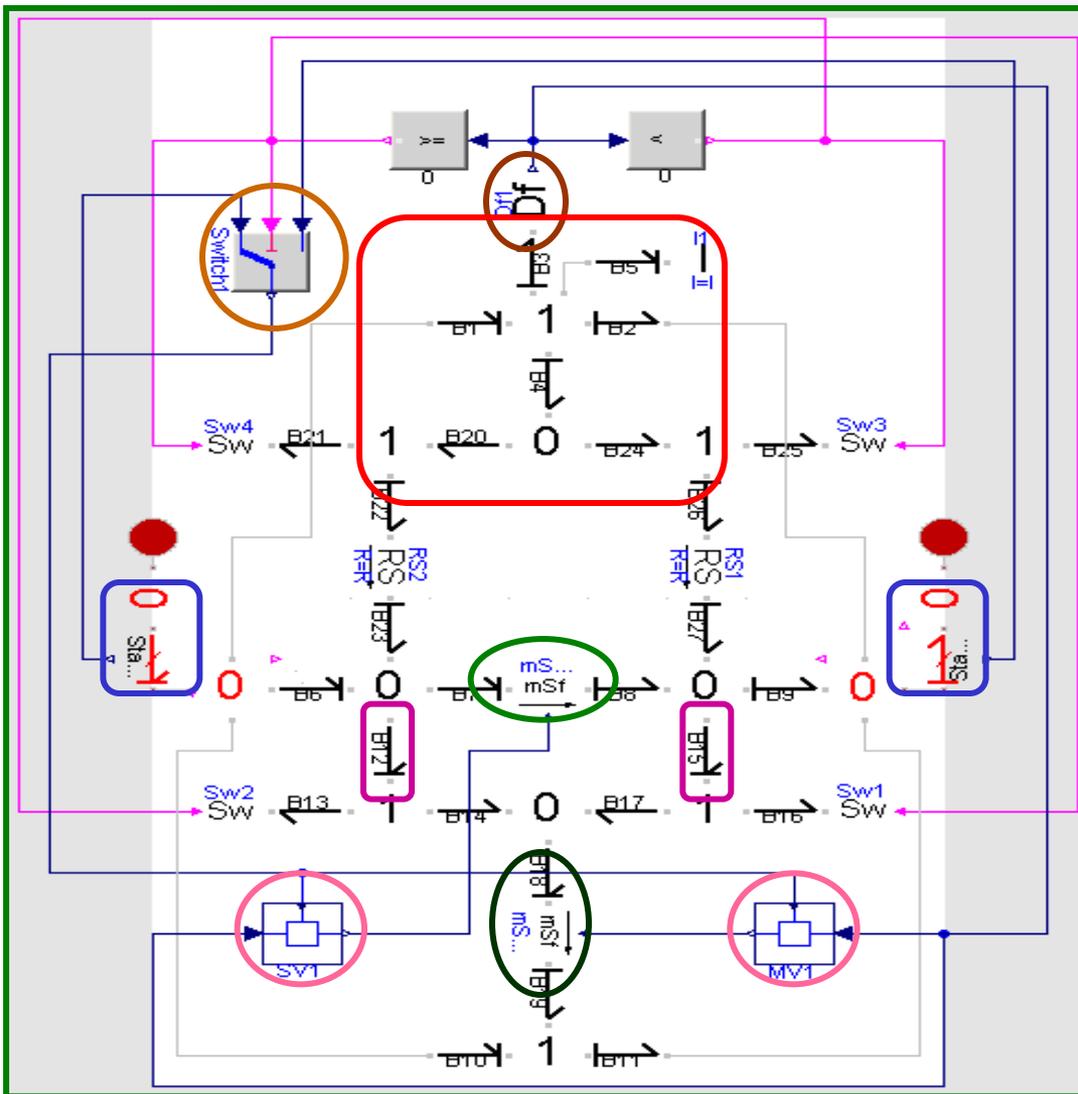


The volume flow is modeled as a wave equation with friction. The friction is in parallel with the inertia.

The flow is measured using a flow sensor element. The additional entropy generated by friction is reintroduced in the down-wind direction, i.e., in the direction of the flow. Switch elements are used to determine the reintroduction point.

Non-linear flow sources are used to model the parallel thermal and mass flows.

These are computed by converting the volume flows to consistent entropy and mass flows.



State sensor elements are used to determine the current values of volume, entropy, and mass.

Up-wind state information is being used to convert the volume flow into consistent entropy and mass flows.

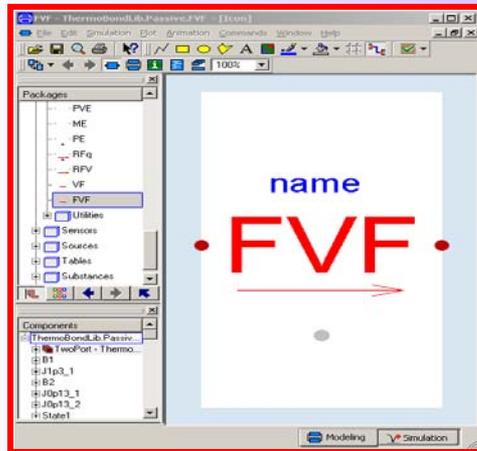
Since entropy doesn't need to be preserved, the non-linear flow source is inserted directly into the thermal branch.

Since mass flow must be preserved, the non-linear flow source is inserted under a 1-junction in the mass flow branch.

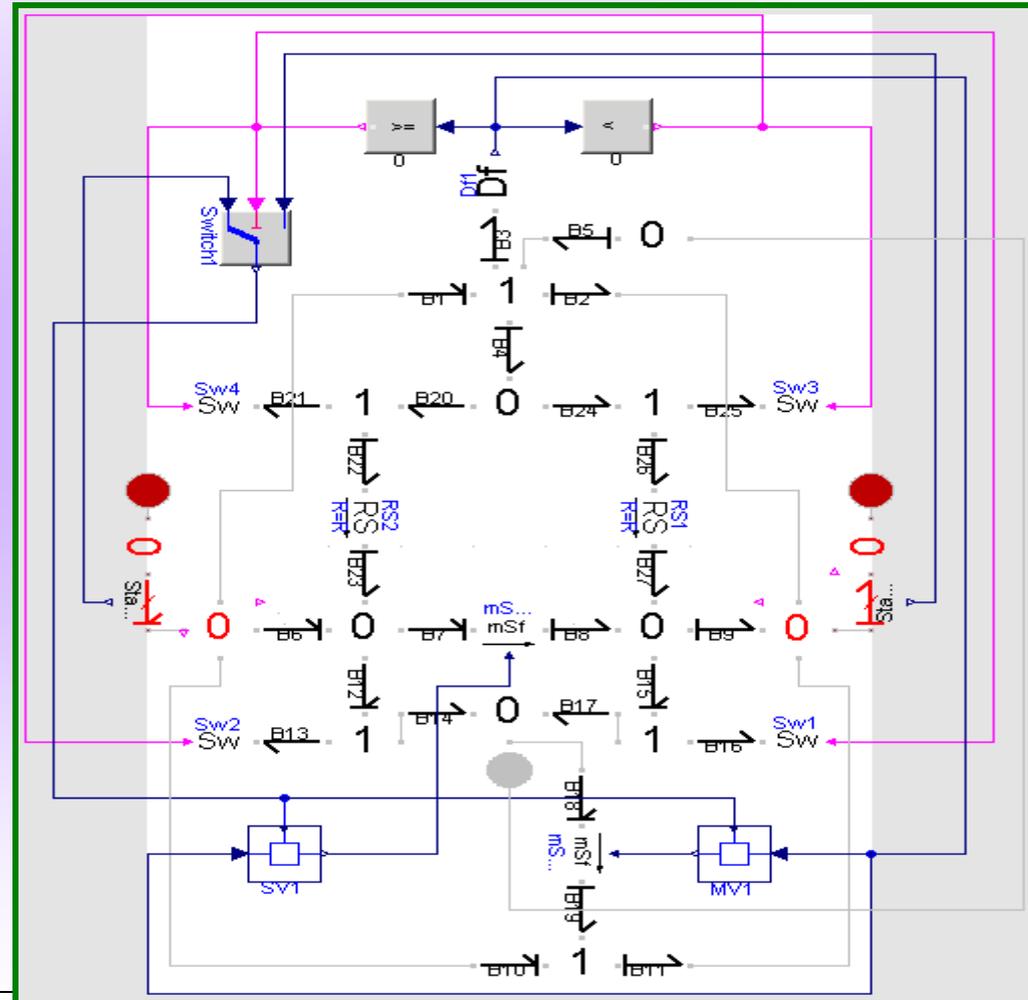
To move mass with the volume, additional energy is needed that is taken from the thermal domain.

Forced Convective Volume Flow

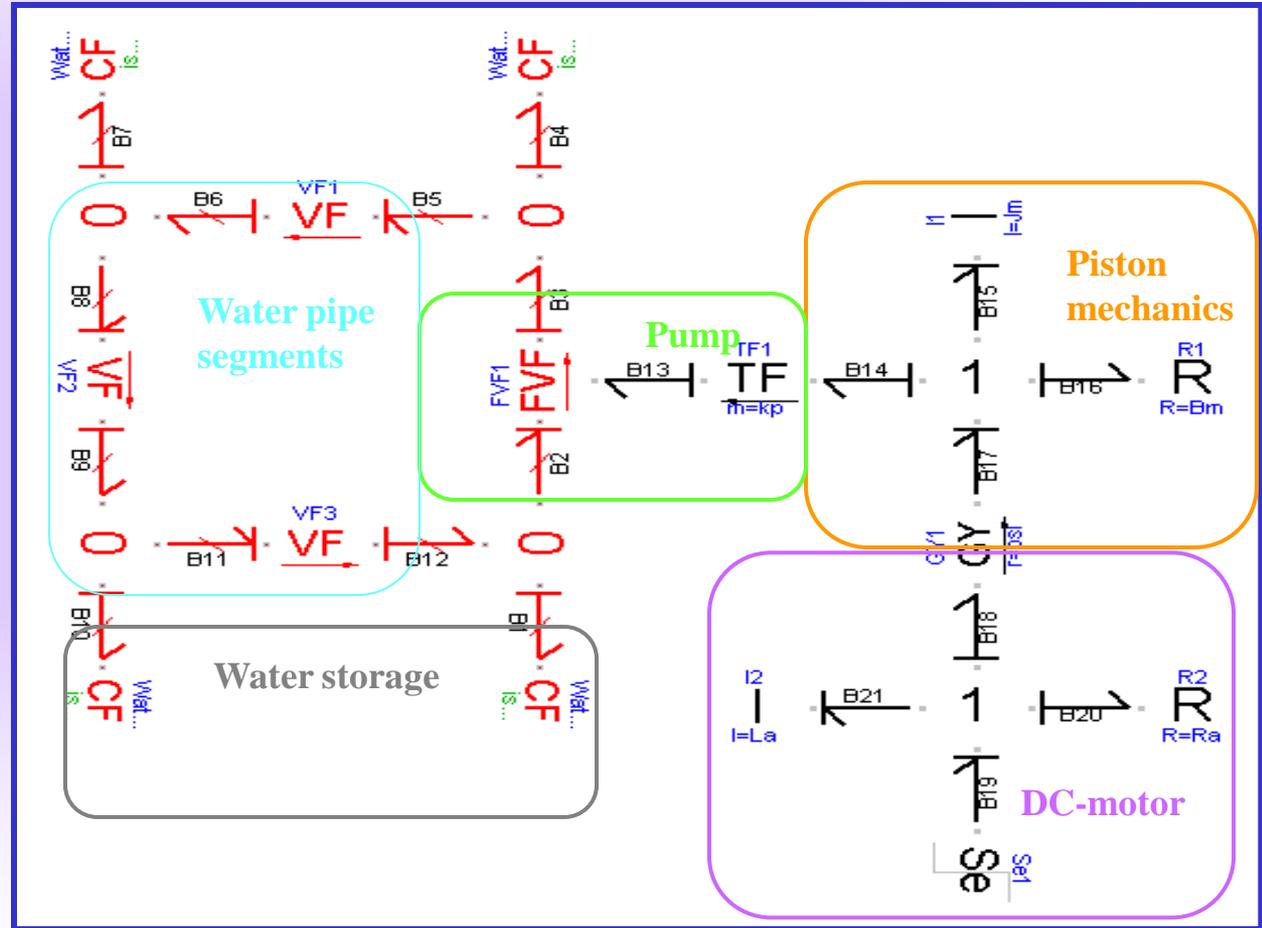
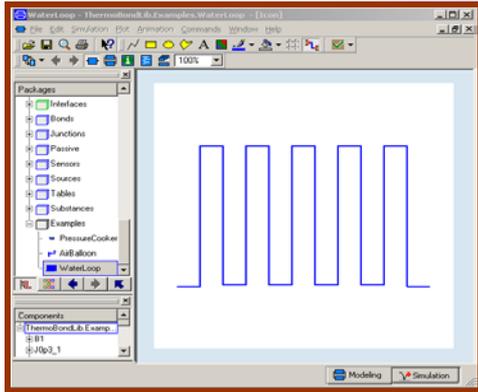
- We are now ready to describe the forced convective mass flow.



The model is almost the same as the free convective flow model, except that a volume flow is forced on the system through the regular bond connector at the top, replacing the inductor.



The Water Serpentine I



The Water Serpentine II

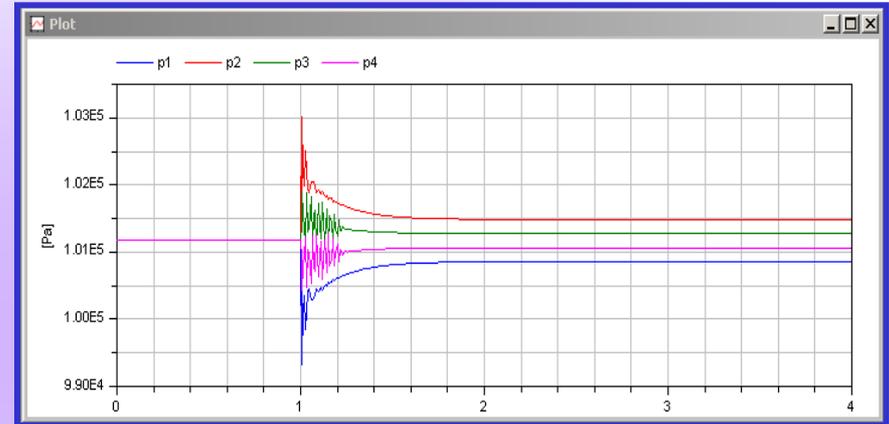
- The pump forces a flow, thereby creating a higher pressure at the outflow, while creating a lower pressure at the inflow.
- Mass is transported through the pump with the volume. Since the mass is getting condensed, it occupies less space. Thus, there is “surplus” volume that gets used to “finance” the mass transport in the *Gibbs equation*.
- In the pipe segments, the pressure is gradually reduced again, thus each pipe segment has a higher pressure at the inflow than at the outflow. The mass thus expands, and the volume consumed in the pump is gradually given back, so that the overall volume in the water serpentine is being preserved.

The Water Serpentine IV

```
Messages - Dymola
Syntax Error Translation Dialog Error Simulation
Log-file of program ./dymosim
(generated: Fri Feb 08 16:13:54 2008)

dymosim started
... "dsin.txt" loading (dymosim input file)
... "WaterLoop.mat" creating (simulation result file)

Integration started at T = 0 using integration method DASSL
(DAE multi-step solver (dassl/dasslrt of Petzold modified by Dynasim))
Integration terminated successfully at T = 4
CPU-time for integration : 0.4 seconds
CPU-time for one GRID interval: 0.8 milli-seconds
Number of result points : 514
Number of GRID points : 501
Number of (successful) steps : 499
Number of F-evaluations : 3336
Number of H-evaluations : 1053
Number of Jacobian-evaluations: 138
Number of (model) time events : 1
Number of (U) time events : 0
Number of state events : 6
Number of step events : 0
Minimum integration stepsize : 6.17e-009
Maximum integration stepsize : 0.476
Maximum integration order : 5
Calling terminal section
... "dsfinal.txt" creating (final states)
```



Comparison With *Biosphere 2*

- In the *Biosphere 2* model, only the (sensible and latent) heat were modeled. The mass flows were not considered.
- Consequently, you never know in the *Biosphere 2* model, how much water is available where. It is always assumed that the pond never dries out, and that the plants always have enough water to be able to evaporate in accordance with their temperature and saturation pressure.
- In the case of the *pressure cooker* model, both the mass flows and the heat flows were modeled and simulated. Consequently, the case is caught, where all the water has evaporated, while the air/steam mixture is still not fully saturated.

References I

- Cellier, F.E. and J. Greifeneder (2003), “Object-oriented modeling of convective flows using the Dymola thermo-bond-graph library,” *Proc. ICBGM’03, Intl. Conference on Bond Graph Modeling and Simulation*, Orlando, FL, pp. 198 – 204.
- Greifeneder, J. and F.E. Cellier (2001), “Modeling multi-element systems using bond graphs,” *Proc. ESS’01, European Simulation Symposium*, Marseille, France, pp. 758 – 766.
- Cellier, F.E. and J. Greifeneder (2008), “ThermoBondLib – A New Modelica Library for Modeling Convective Flows,” *Proc. Modelica’08*, Bielefeld, Germany, pp. 163 – 172.

References II

- Cellier, F.E. (2007), *The Dymola Bond-Graph Library*, Version 2.3.
- Cellier, F.E. (2007), *The Dymola Thermo-Bond-Graph Library*, Version 2.0.

