Important Characteristics of VHDL-AMS and Modelica with Respect to Model Exchange

1st International Workshop on Equation-Based Object-Oriented Languages and Tools, Berlin, July 30, 2007

Olaf Enge-Rosenblatt,
Joachim Haase,
Christoph Clauß

Fraunhofer Institute for Integrated Circuits
Design Automation Division
Dresden, Germany

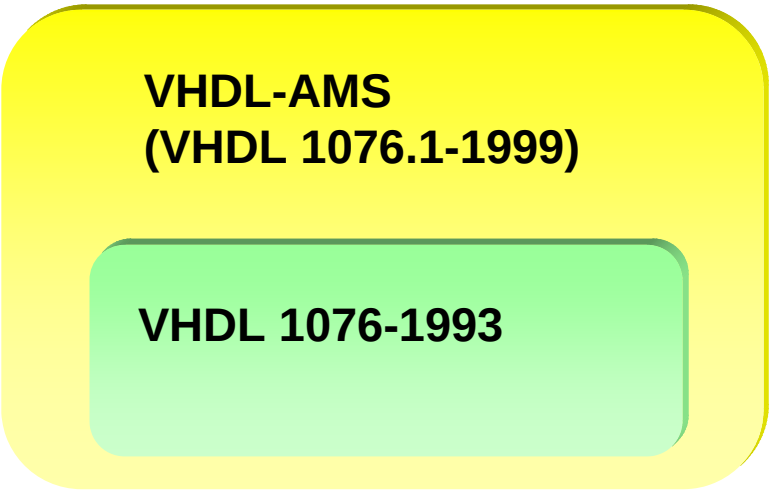
Fraunhofer Institut Integrierte Schaltungen

Outline

1. **Language characteristics**
   - VHDL-AMS
   - Modelica
2. **Examples**
   - Conservative system
   - Structural description
3. **Comparision of some aspects**
4. **Transformation of models**
   - Modelica ➔ VHDL-AMS
   - VHDL-AMS ➔ Modelica
5. **Conclusions**

Fraunhofer Institut Integrierte Schaltungen

VHDL-AMS



- VHDL 1076-1993
  - Time-discrete systems
  - Event-driven simulation algorithms
- VHDL-AMS
  - Extension to time-continuous (analog) systems
  - Analog DAE-solver for differential algebraic systems of equations

$$F(x, \dot{x}, t) = 0$$

  - IEEE standard 1076.1-1999 (revised in 2007)
  - http://www.vhdl.org/analog

VHDL-AMS is a superset of VHDL.
Concepts and language constructs of digital VHDL remain valid.

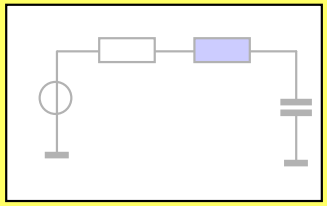
Modelica

- Modelica
  - Object-oriented modelling language
  - Description of physical systems based on
    - differential equations
    - algebraic equations
    - discrete equations
  - Modelica Association
    Modelica Language Specification
    Version 2.0, July 10, 2002
  - http://www.modelica.org

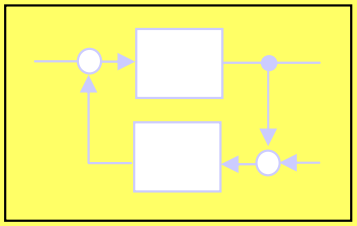
Modelica is a unified language specification for systems characterized by differential-algebraic equations taking into account discrete events for definition of discontinuous behaviour or sampled-signal systems.

Modelling approach

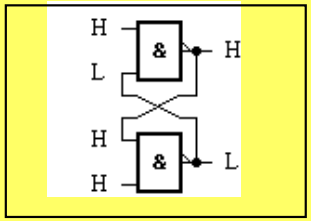
Mathematical model



conservative systems
(Kirchhoff's network)



non-conservative systems
(signal flow graph)



digital systems
(time-discrete)

mixed description possible
- analog-digital
- electrical – non-electrical
- conservative – non-conservative
- ...

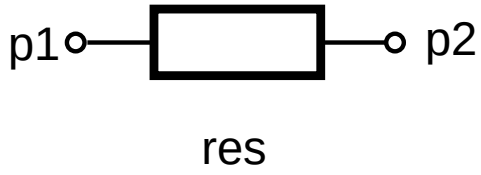
Implementation of models

VHDL-AMS

Modelica

Fraunhofer Institut Integrierte Schaltungen

Resistance model in VHDL-AMS (1)

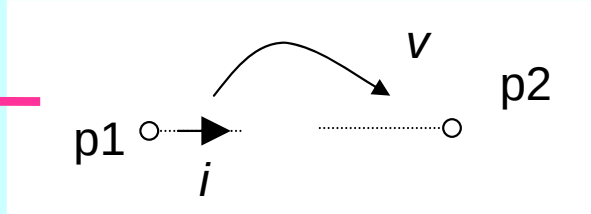



```
library IEEE;
use IEEE.electrical_systems.all;
```

```
entity resistor is
  generic ( res  : resistance);          -- Resistance [Ohm]
  port    ( terminal p1, p2 : electrical);
end entity resistor;
```

one instance (obj.-orient.)

```
architecture ideal of resistor is
  quantity v across i through p1 to p2;

begin
  v == res * i;

end architecture ideal;
```



$$v = res \cdot i$$

Fraunhofer Institut Integrierte Schaltungen

Resistance model in VHDL-AMS (2)

```
-- library IEEE_proposed
package ELECTRICAL_SYSTEMS is

-- subtype declarations
   subtype VOLTAGE is REAL tolerance "DEFAULT_VOLTAGE";
   subtype CURRENT is REAL tolerance "DEFAULT_CURRENT";

-- nature declarations
   nature ELECTRICAL is
          VOLTAGE        across
          CURRENT        through
          ELECTRICAL_REF reference;

-- ...
end package ELECTRICAL_SYSTEMS;
```

ELECTRICAL – **nature** name to determine physical domain

CURRENT – type of the associated flow quantity

VOLTAGE – type of the associated non-flow quantity

ELECTRICAL_REF – identifier of the electrical reference node (potential = 0.0)

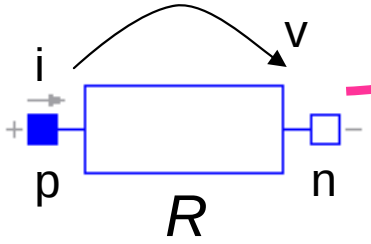
Resistance model in Modelica (1)

physical domain

```
model Resistor "Ideal linear electrical resistor"
  extends Modelica.Electrical.Analog.Interfaces.OnePort;

  parameter SIunits.Resistance R=1 „Resistance";
```

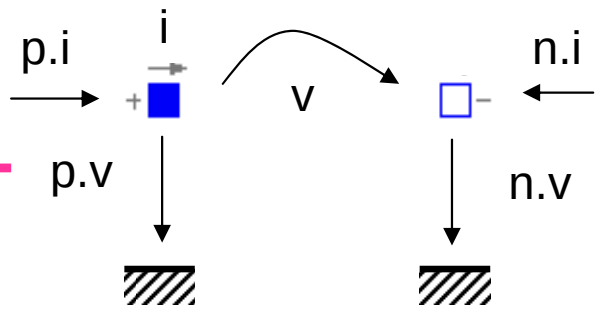
```
equation
  v = R*i;

end Capacitor;
```

$$v = R \cdot i$$



super class (obj.-orient.)

```
partial model OnePort
  "Component with two electrical pins p and n and current i from p to n"
  SIunits.Voltage v  "Voltage drop between the two pins (= p.v - n.v)";
  SIunits.Current i  "Current flowing from pin p to pin n";
  Modelica.Electrical.Analog.Interfaces.PositivePin p;
  Modelica.Electrical.Analog.Interfaces.NegativePin n;

equation
  v = p.v - n.v;
  0 = p.i + n.i;
  i = p.i;
end OnePort;
```



Fraunhofer Institut Integrierte Schaltungen

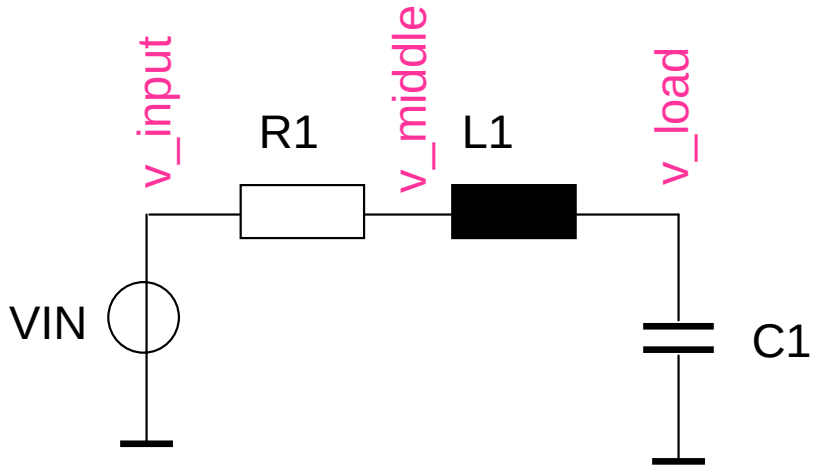
Resistance model in Modelica (2)

```
// Modelica.Electrical.Analog.Interfaces

connector PositivePin "Positive pin of an electric component"
  SIunits.Voltage      v "Potential at the pin";
  flow SIunits.Current i "Current flowing into the pin";
end PositivePin;

connector NegativePin "Negative pin of an electric component"
  SIunits.Voltage      v "Potential at the pin";
  flow SIunits.Current i "Current flowing into the pin";
end NegativePin;
```


```
// Modelica.SIunits

type ElectricCurrent = Real (final quantity="ElectricCurrent", final unit="A");
type Current = ElectricCurrent ;

type ElectricPotential = Real (final quantity="ElectricPotential", final unit="V");
type Voltage = ElectricPotential ;
```


Fraunhofer Institut Integrierte Schaltungen

RLC Circuit in VHDL-AMS




```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.electrical_systems.all;
```

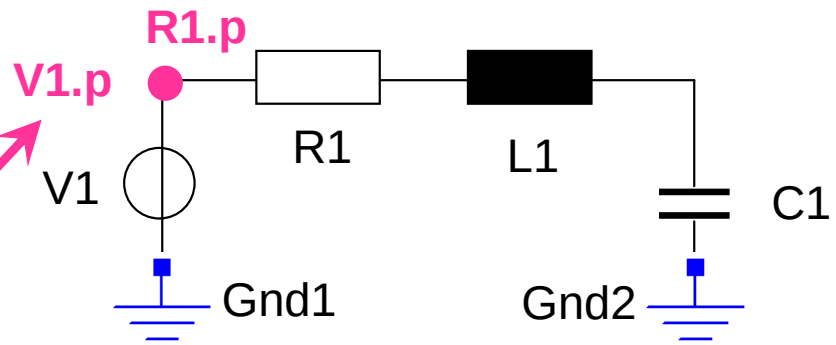
```
entity RLC is end RLC;
```

```
architecture ex_rlc of RLC is
    terminal v_input, v_middle, v_load : electrical;
begin
 VIN : entity work.v_ramp(ideal)
             generic map( pulse => 5.0, duration => 10.0e-6)
             port map(pos => v_input, neg => ELECTRICAL_REF);

 R1 :  entity work.resistor(ideal) generic map(res => 3.0)
             port map(p1 => v_input, p2 => v_middle);

 L1 :  entity work.inductor(ideal) generic map(ind => 1.0e-3)
             port map(p1 => v_middle, p2 => v_load);

 C1 :  entity work.capacitor(ideal) generic map(cap => 10.0e-6)
             port map(p1 => v_load, p2 => ELECTRICAL_REF);

end architecture ex_rlc;
```

Instantiation

- parameter assignment (generic map)
- terminal assignment (port map)

RLC Circuit in Modelica (1)



used libraries

```
model RLC
  Modelica.Electrical.Analog.Sources.RampVoltage V1
                                   (V=5,duration=10.0e-6);
  Modelica.Electrical.Analog.Basic.Resistor  R1(R=3.0);
  Modelica.Electrical.Analog.Basic.Capacitor C1
                                   (C=10.0e-6, v(start=-5));
  Modelica.Electrical.Analog.Basic.Inductor  L1(L=1.0e-3);
  Modelica.Electrical.Analog.Basic.Ground    Gnd1;
  Modelica.Electrical.Analog.Basic.Ground    Gnd2;

equation
  connect(V1.n, Gnd1.p);
  connect(V1.p, R1.p);
  connect(R1.n, L1.p);
  connect(L1.n, C1.p);
  connect(C1.n, Gnd2.p);

end RLC;
```

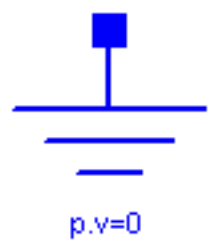
super classes

instances with their parameters

description of pin connection

RLC Circuit in Modelica (2)

```
// Modelica.Electrical.Analog.Basic

model Ground "Ground node"
  Modelica.Electrical.Analog.Interfaces.Pin p;
equation
  p.v = 0;
end Ground;
```



Modelica Language Specification:

connect (a, b)
a.across = b.across
a.through + b.through = 0

connect (a, b)
connect (a, c)
a.across = b.across = c.across
a.through + b.through + c.through = 0

Fraunhofer Institut Integrierte Schaltungen

Comparison (1)

Aspect	VHDL-AMS	Modelica
Definition	IEEE Std. 1076.1 (revised 2007)	Modelica Specification 2.2 Modelica Association
Time-continuous modeling	conservative (networks) non-conservative (signal-flow)	physical modeling block-oriented modeling
Time-discrete modeling	event-driven	event-driven no event queue
Model interface	**entity**	**model** **block**
Model parameter	**generic** parameter	**parameter**
Connection points	**port** (**terminal**, **quantity**, **signal**)	specified by connector classes
Connection point characterization	**nature** for terminals (**through**, **across**, **reference**)	connector (**flow**, non-flow)

Fraunhofer Institut Integrierte Schaltungen

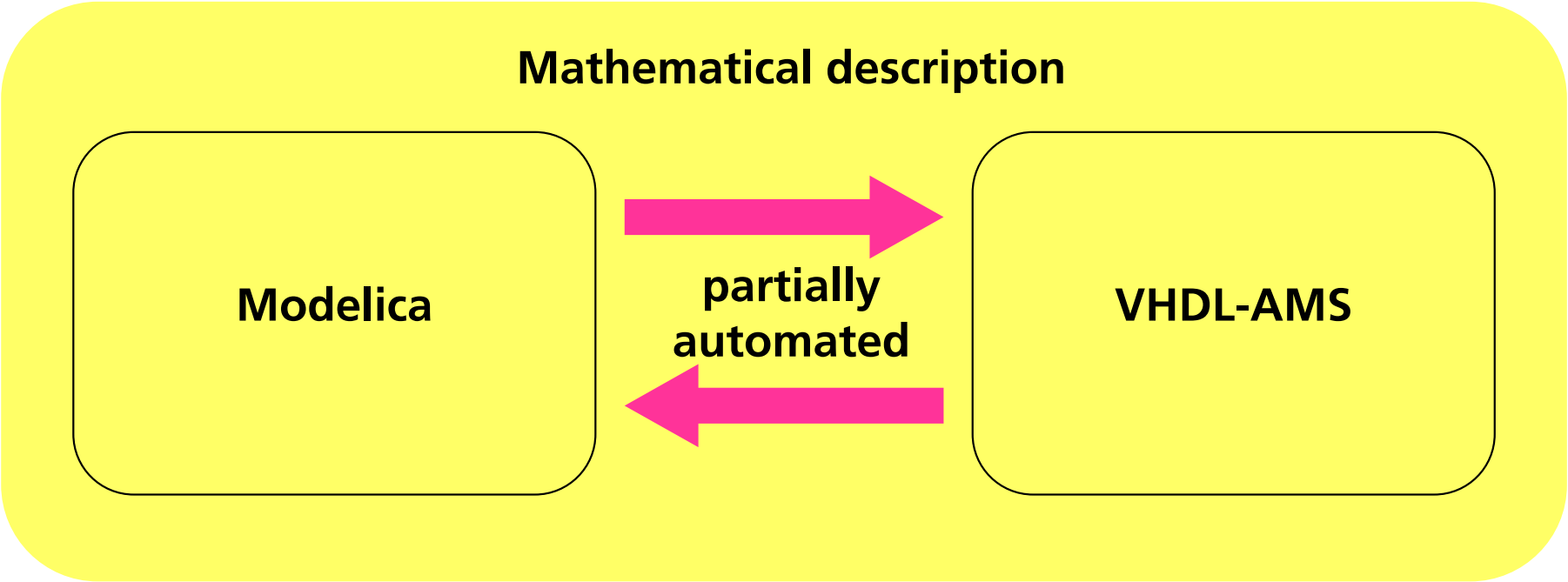
Comparison (2)

Aspect	VHDL-AMS	Modelica
Model behavior	**architecture**	**equation** part, **algorithm**
Organisation	one or more architectures corresponding to one entity	different models for different levels of abstraction
Analog behavior	equation oriented expression1 == expression2;	equation oriented expression1 = expression 2;
Analog waveform	**quantity**	**dynamic variable**
Initial conditions	**break** statement	**initial equation** **fixed start** values
Discontinuities	**break** statement	**reinit**();
Vector operations	overloading of operators	built-in functions

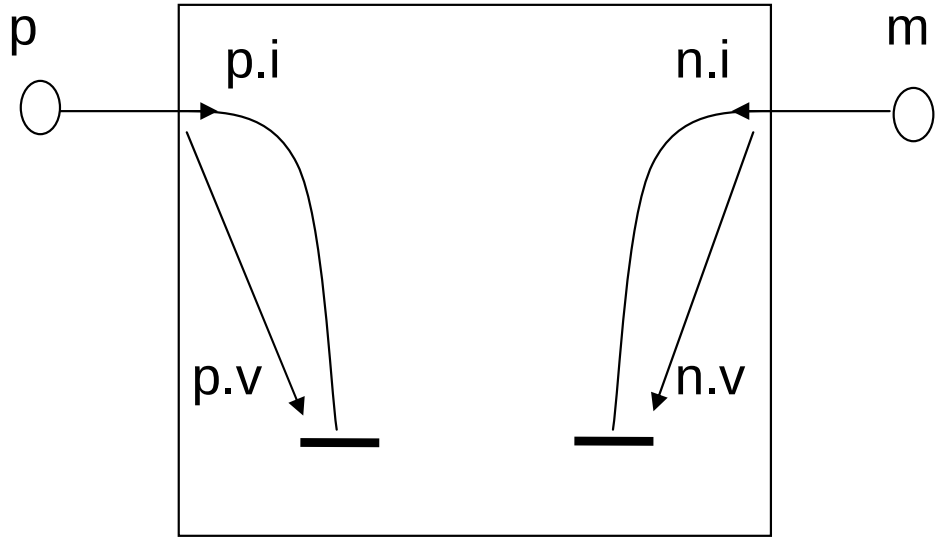
Comparison (3)

Aspect	VHDL-AMS	Modelica
Digital waveform	**signal**	**discrete** (Pin)
D/A conversion	‘RAMP, ‘SLEW	**smooth** ()
Inheritance	Not supported	Widely used
Netlists	Instance oriented	Connection point oriented

Model exchange



Underlying Branch Structure in Modelica



Basis for transformation of conservative model from Modelica to VHDL-AMS

Fraunhofer Institut Integrierte Schaltungen

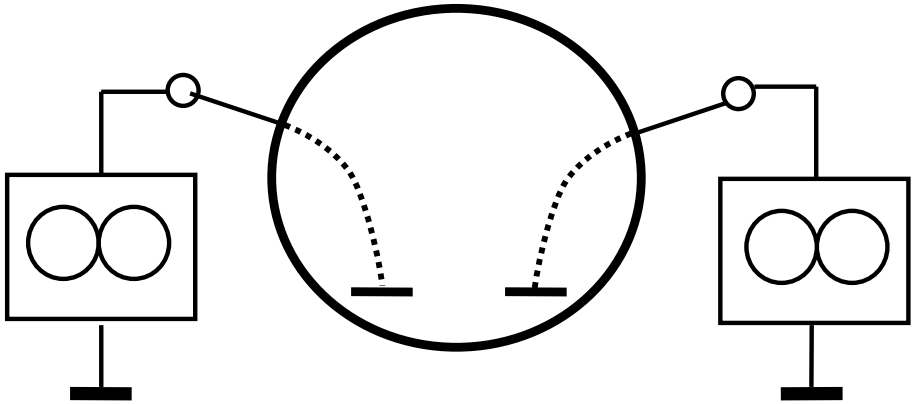
Modelica to VHDL-AMS (example Resistor)

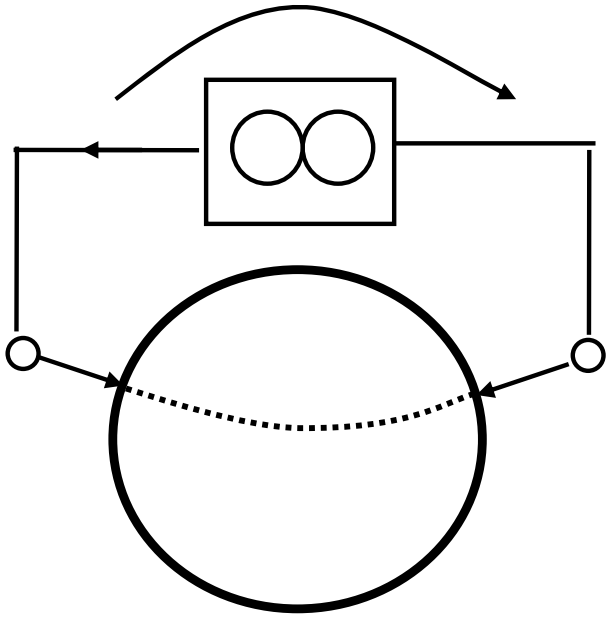

```
partial model OnePort
   "Component with two pins
  ….Voltage v  "p.v - n.v";
  ….Current i  "from p to n";
  ….PositivePin p;
  ….NegativePin n;
equation
  v = p.v - n.v;
  0 = p.i + n.i;
  i = p.i;
end OnePort;

model Resistor
  "Ideal resistor"
extends
  ….OnePort;
  parameter ….Resistance R=1;
equation
  v = R*i;
end Resistor;
```

```
library IEEE;
use IEEE.ELECTRICAL_SYSTEMS.all;
entity RESISTOR is
  generic(R : RESISTANCE := 1.0);
  port   (terminal P: ELECTRICAL;
          terminal N: ELECTRICAL);
end entity RESISTOR;
architecture MODELICA of RESISTOR is
  quantity P_V across P_I through P;
  quantity N_V across N_I through N;
  quantity V   : REAL;
  quantity I   : REAL;
begin
  V   == P_V – N_V;
  0.0 == P_I + N_I;
  I   == P_I;
  V   == R*I;
end architecture MODELICA;
```

Model correct, but …

Establishing Small (Smart) Models

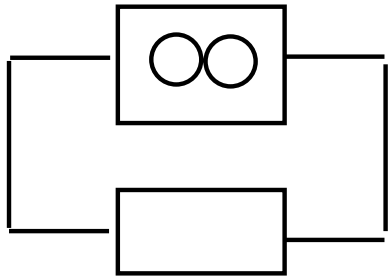




- group terminals
- connect terminals with norators
- establish (and reduce) network equations

Fraunhofer Institut Integrierte Schaltungen

Example - Resistor (1)



$$\begin{pmatrix} 1 & -1 & . & . & -1 & . & . & . \\ . & . & 1 & 1 & . & . & . & . \\ . & . & 1 & . & . & -1 & . & . \\ . & . & . & . & 1 & -R & . & . \\ 1 & -1 & . & . & . & . & -1 & . \\ . & . & 1 & . & . & . & . & 1 \\ . & . & . & 1 & . & . & . & -1 \end{pmatrix} \cdot \begin{pmatrix} p.v \\ n.v \\ p.i \\ n.i \\ v \\ i \\ V_{NOR} \\ I_{NOR} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Reduction step

$$\begin{pmatrix} 1 & -1 & . & . & . & . & . & -R \\ . & . & 1 & . & . & . & . & 1 \\ . & . & . & 1 & . & . & . & -1 \\ . & . & . & . & 1 & . & . & R \\ . & . & . & . & . & 1 & . & 1 \\ . & . & . & . & . & . & 1 & -R \\ . & . & . & . & . & . & . & . \end{pmatrix} \cdot \begin{pmatrix} p.v \\ n.v \\ p.i \\ n.i \\ v \\ i \\ V_{NOR} \\ I_{NOR} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

smart model
- structure eq. norator structure
- simultaneous statements from (reduced) equations

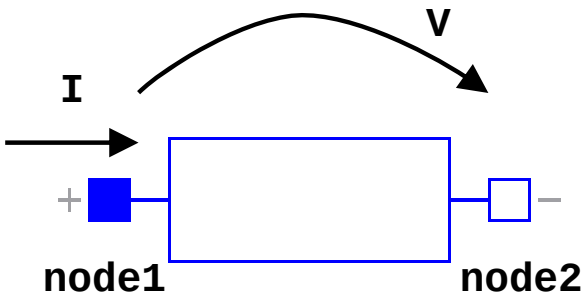
Example - Resistor (2)

```
architecture IDEAL of RESISTOR is
  quantity V_NOR across I_NOR through P to N;
begin
  V_NOR == R*I_NOR;
end architecture IDEAL;
```

Only one branch and one simultaneous statement

Example - Resistor

```
architecture R1 of RESISTOR is
  quantity V across
           I through node1 to node2;
begin
  V == R*I;
end architecture R1;
```




```
connector vhdl_pin
  SIunits.Voltage      v;
  flow SIunits.Current i;
end vhdl_pin;
vhdl_pin node1, node2;
```

```
model Resistor
  extends ….OnePort;
equation
  v = R*i;
end Resistor;
Resistor R1;
...
equation
  connect(R1.p, node1);
  connect(R1.n, node2);
...
```

Conclusions

- Same mathematical modelling approaches are supported by VHDL-AMS and Modelica

- Differences result from the semantic approach:
  - VHDL-AMS comes from electronics
  - Modelica deals mainly with multi-physics problems

- Potential to transform models from one language into the other
  - Special underlying modelling approaches should be considered

- **Transformation of conservative models from Modelica to VHDL-AMS would be easier if access to the (reduced) model equations would be possible!**

Fraunhofer Institut Integrierte Schaltungen

Thank You!

Fraunhofer Institut
Integrierte Schaltungen