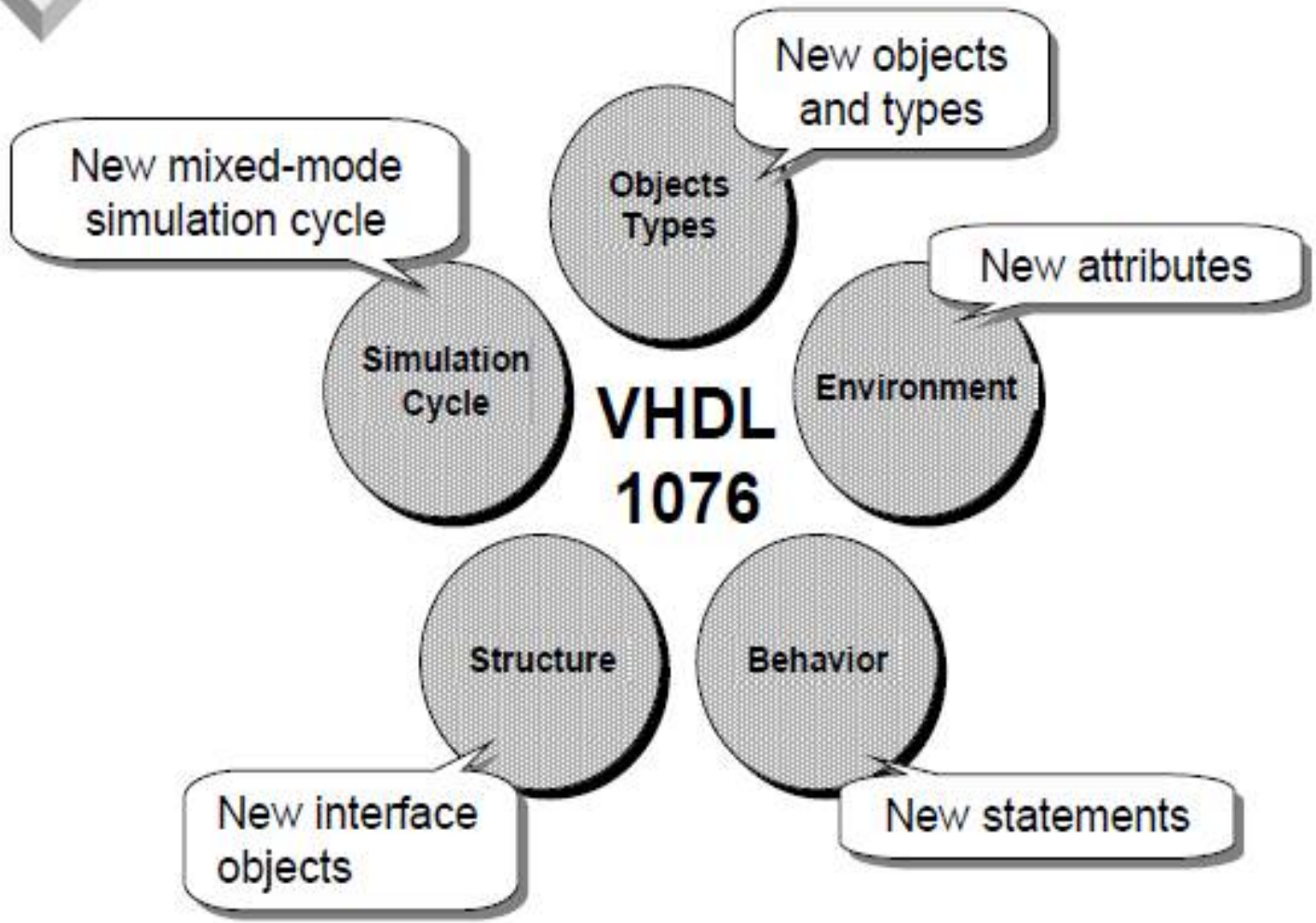
VHDL-AMS

Why is VHDL-AMS needed?

- VHDL 1076 is suitable for modeling and simulating discrete systems
- Many of today's designs include at least some continuous characteristics:
  - System design
    Mixed-signal electrical designs
    Mixed electrical/non-electrical designs
    Modeling design environment
  - Analog design
    Analog behavioral modeling and simulation
  - Digital design
    Detailed modeling (e.g. submicron effects)
- Designers want a uniform description language

VHDL AMS

VHDL-AMS Language Architecture



VHDL-AMS Concepts

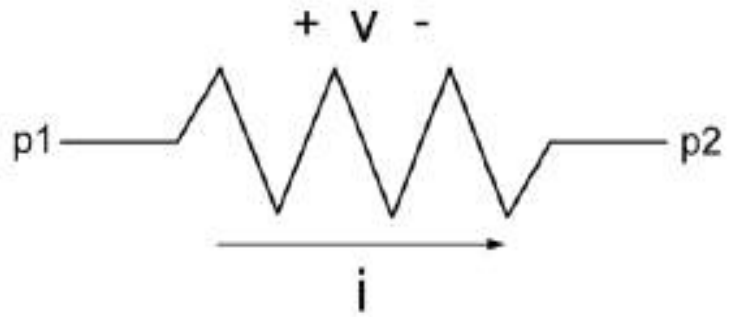
- VHDL-AMS models are organized as **entities** and **architectures**
- It has a concept of **time, concurrent processes**
- It has a **well-defined simulation cycle**
- It can model **continuous and discontinuous** behavior
- Equations are solved using **conservation laws** (e.g. KCL, Newton's Laws)
- It handles **initial conditions**, **piecewise-defined behavior**, and so forth

VHDL-AMS Model Structure

VHDL-AMS models are typically comprised of two sections: an **entity** and an **architecture**.

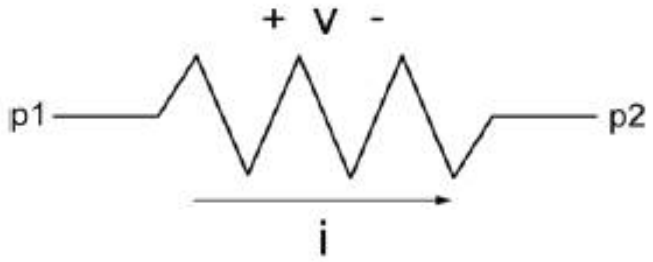
- Entity - Describes the model **interface** to the outside world
- Architecture - Describes the **function** or **behavior** of the model

Entity - model interface



- Pins "p1" and "p2" provide the **interface** between this model and the outside world.
- The nature of these pins is defined in the model's "Entity declaration."

Resistor Model (Entity Declaration)

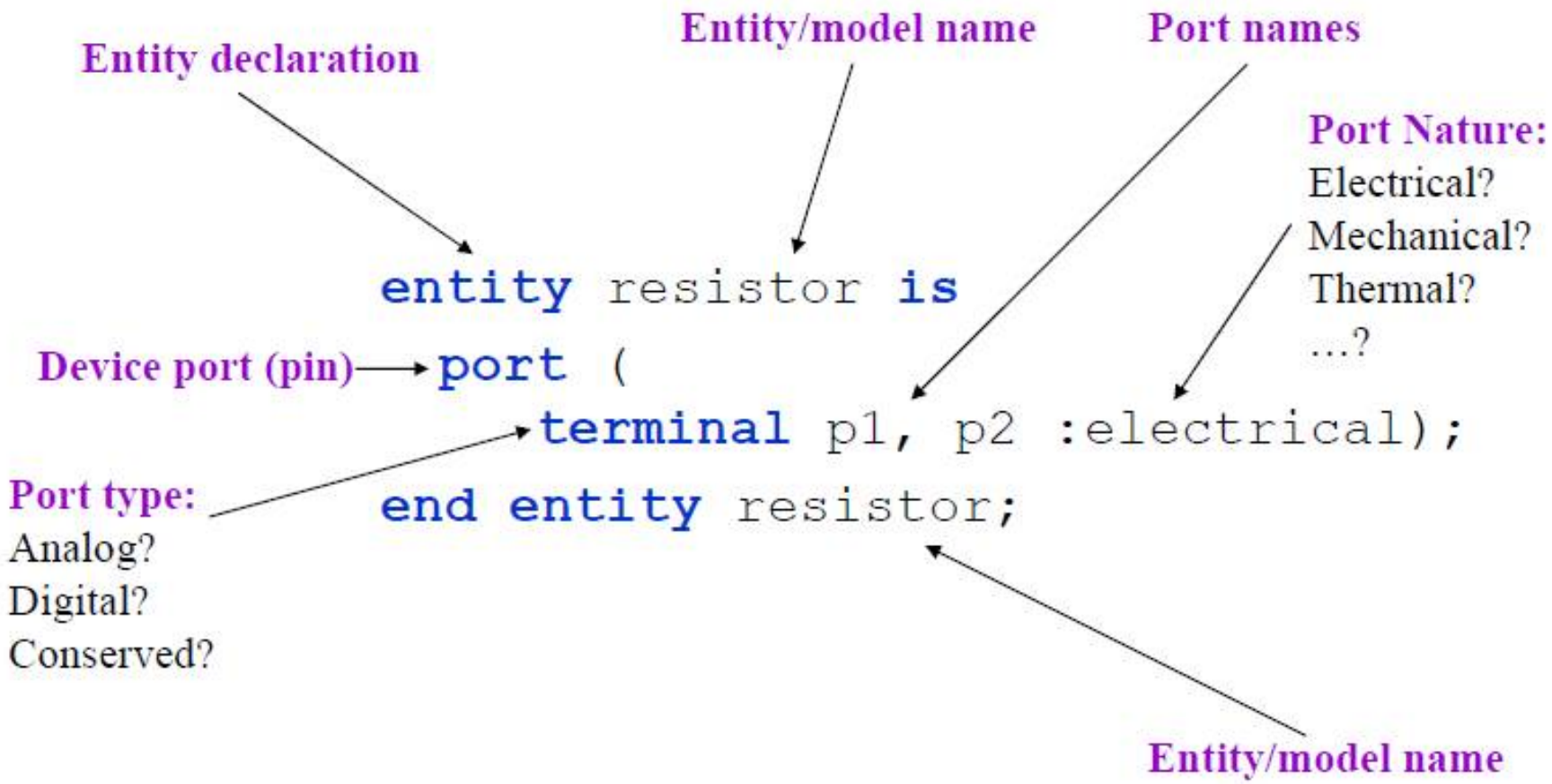


Pin Definitions:

p1, p2 - electrical pins

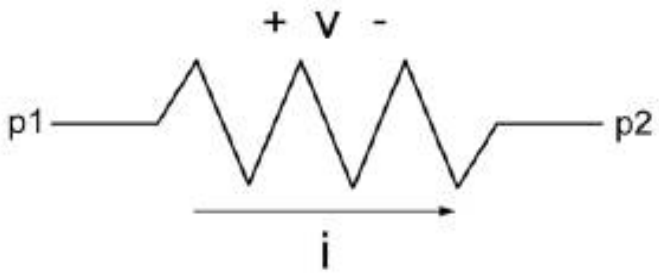
```
entity resistor is
  port (
     terminal  p1, p2 : electrical);
end entity resistor;
```

Resistor Model (Entity Explanation)



Architecture - model behavior

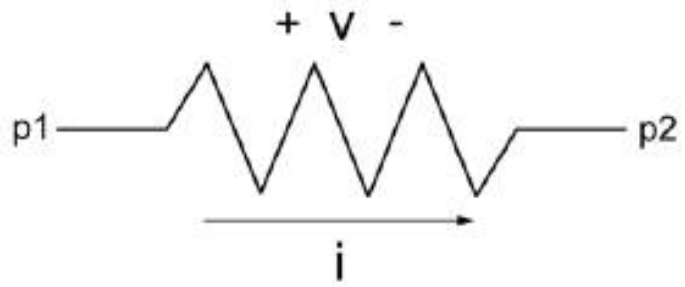
- The **architecture** describes the **behavior** of the model.



- In this case, the model behavior is governed by Ohm's law, which relates current and voltage as:

$$i = v / res$$

Resistor Model (Architecture)

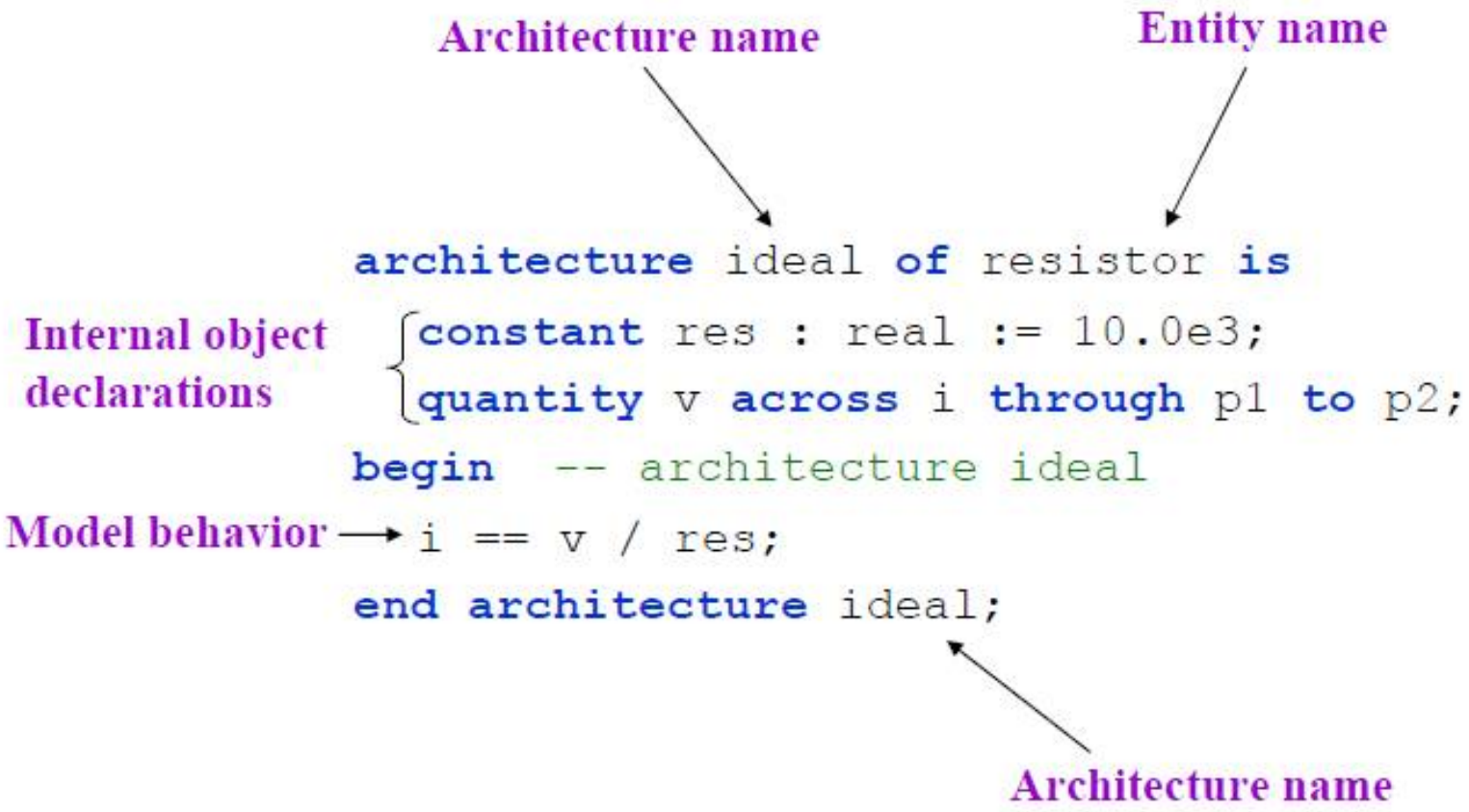


Characteristic Equation:

$i = v / res$

```
architecture ideal of resistor is
  constant res : real := 10.0e3;
  quantity v across i through p1 to p2;
begin  -- architecture ideal
  i == v / res;
end architecture ideal;
```

Resistor Model (Architecture Explanation)



VHDL-AMS Object Types

- There are six classes of "objects" in VHDL-AMS:
  - Constants
  - Terminals
  - Quantities
  - Variables
  - Signals
  - Files
- For analog modeling, **constants**, **terminals**, and **quantities** are routinely used

Constants

- **Data storage** object for use in a model
  - **constant** res : real := 50.0;

    Declares constant, **res**, of type *real*, and initializes it to 50.0. Since this constant is of type real, it must be assigned only real values, which must include a decimal point.
  - **constant** count : integer := 3;

    Declares constant **count**, of type *integer*, and initializes it to 3. Since count is of type integer, it must be assigned only whole values, which must not include a decimal point.
  - **constant** td : time := 1 ns;

    Declares constant **td**, of type *time*, and initializes it to 1 ns (1.0e-9 seconds).

    Time is a special kind of constant, described next.

Predefined Physical Types

- The constant time is a *predefined physical type*, so named because it represents a real world physical property. It can be either real or integer.
- As a physical type, time values are specified with a value followed by a multiplier (separated with a space). Predefined time multipliers consist of the following:

- **fs**	**(femto-seconds)**
- **ps**	**(pico-seconds)**
- **ns**	**(nano-seconds)**
- **us**	**(micro-seconds)**
- **ms**	**(milli-seconds)**
- **sec**	**(seconds)**
- **min**	**(minutes)**
- **hr**	**(hours)**

Terminals

- Terminals represent **continuous, conservative ports** in VHDL-AMS
- Terminals have **across** (potential) and **through** (flow) **aspects**
- Terminal types are referred to as "**natures**"

Quantities:

- **Free quantity** - non-conservative analog object:
  - `quantity pwr : real;`
- **Branch quantity** - analog object used for conservative energy systems:
  - `quantity v across i through p1 to p2;`

Branch Quantities

- **Branch quantity**

  Branch quantities are analog objects used for conservative energy systems. For electrical systems, these quantities are used to access either the voltage or current, or both, of a terminal port.

  To illustrate branch quantities, consider the entity declaration for the resistor model discussed previously:

- `terminal` p1, p2 : electrical;

Branch Quantities

Quantity "i" refers to the **through** aspect of terminal ports p1 and p2

```
quantity v across i through p1 to p2 ;
```

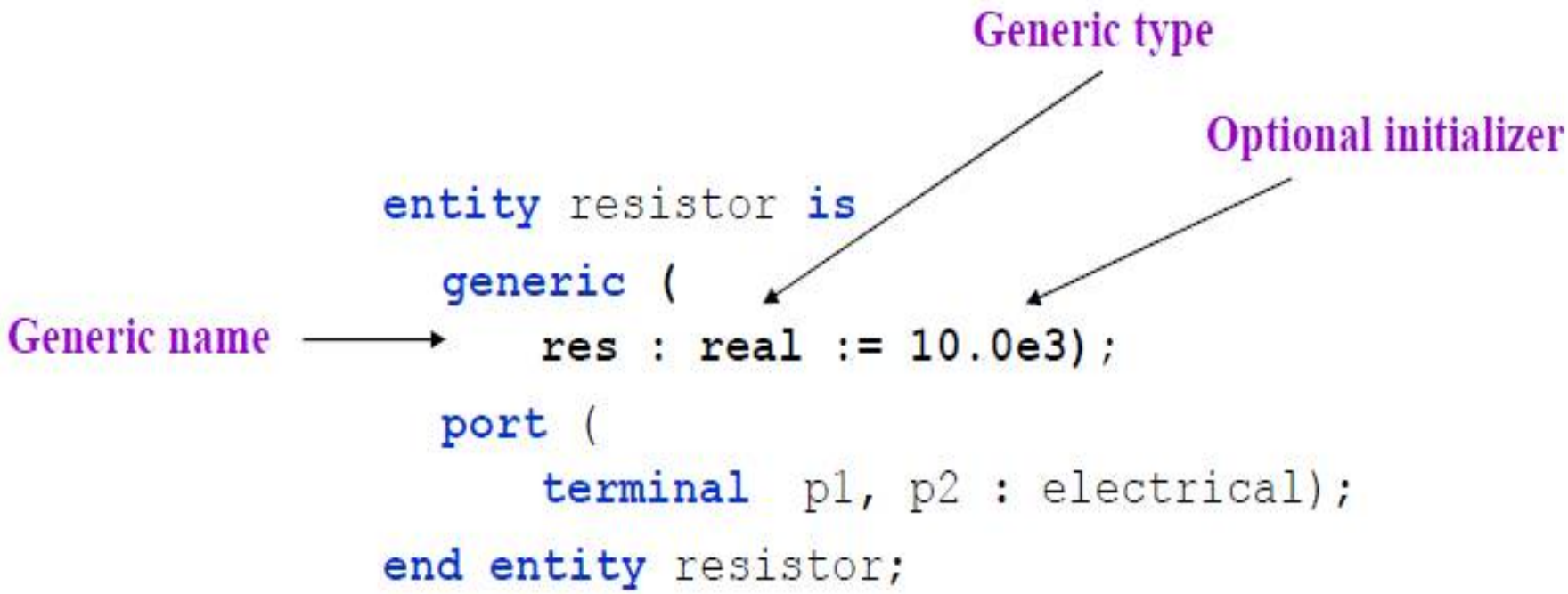
Quantity "v" refers to the **across** aspect of terminal ports p1 and p2

"v" and "i" are defined with respect to terminal ports p1 and p2

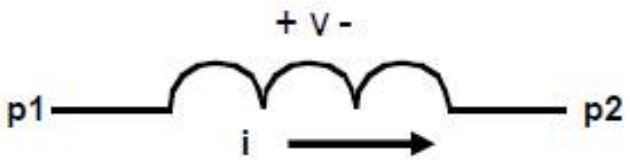
Generic Constants ("Generics")

- Allow models to be **externally parameterized**
- Static objects can be defined as **generics in the entity** of a model, rather than as constants in the architecture of a model
- Allows the model to be used more "generically," **without having to modify the model** itself. The model user just passes in a value to the model.

Resistor Model (Entity with Generic)



Inductor Model (Entity)

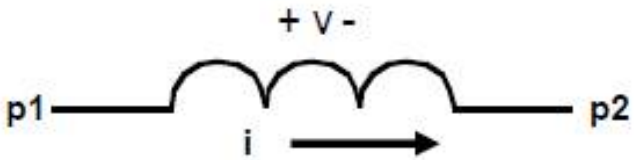


Pin Definitions/Argument:

p1, p2 : electrical pins
ind : user supplied argument

```
use ieee.electrical_systems.all;
entity inductor is
  generic (
      ind : real);     -- inductance value
  port (
       terminal p1, p2 : electrical);
end entity inductor;
```

Inductor Model (Architecture)

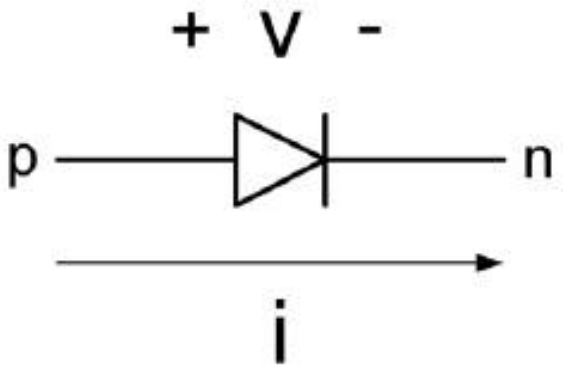


Fundamental Equation:

$$v = ind \frac{di}{dt}$$

```
architecture ideal of inductor is
  quantity v across i through p1 to p2;
begin  -- ideal architecture
  v == ind * i'dot;
end architecture ideal;
```

Diode Model (Entity)

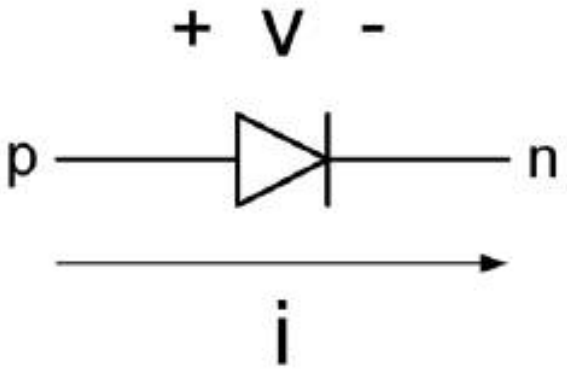


Pin Definitions/Argument:

p, n : electrical pins
Isat : user supplied argument

```
entity diode is
  generic (
    -- saturation current
    Isat : current := 1.0e-14;
  port (
    terminal p, n : electrical);
end entity diode;
```

Diode Model (Architecture)



Fundamental Equation:

$$i = Isat * (\exp^{\frac{v}{vt}} - 1.0)$$

```
architecture ideal of diode is
  constant TempC : real := 27.0;
  constant TempK : real := 273.0 + TempC;
  constant vt : real := PHYS_K*TempK/PHYS_Q;
  quantity v across i through p to n;
begin
  i == Isat*(exp(v/vt)-1.0);
end architecture ideal;
```