IL2204 VHDL-AMS lecture 2

- Attributes
- Discontinuities
- Mixed signals
- Solvability
- Examples

From last lecture

- Nodes are represented by terminals.
- A terminal is of a specified nature, which defines the continuous values associated with the terminal.
- The nature represents the energy domain for the terminal.

- Quantities
  - Free quantities
  - Branch quantities
  - Source quantities

Attributes

Attributes is used to find information about types, natures and signals.

Attributes of Terminals

T'Reference

An across quantity (e.g. voltage to ground)

T'Contribution

Value equals the sum of the values of all through quantities incident to T (with the appropriate sign)

Attributes of Quantities

Attributes for quantity Q

Q'tolerance
Q'above(VALUE)
Q'delayed(TIME)
Q'dot
Q'integ
Q'slew(MAX_RISING_SLOPE, MAX_FALLING_SLOPE)
Q'zt(NUM, DEN, T, INITIAL_DELAY)
Q'ltf(NUM, DEN)
Q'zoh(T, INITIAL_DELAY)

Example – what is this?

```
library ieee;  use ieee.std_logic_1164.all;
library ieee_proposed;  use ieee_proposed.electrical_systems.all;

entity comparator is
  port ( terminal a : electrical;
         signal d : out std_ulogic );
end entity comparator;

------------------------------------------------------------

architecture ideal of comparator is
  constant ref_voltage : real := 5.0;
  quantity vin across a;
begin

  comparator_behavior : process is
  begin
    if vin > ref_voltage / 2.0 then
      d <= '1' after 5 ns;
    else
      d <= '0' after 5 ns;
    end if;
    wait on vin'above(ref_voltage / 2.0);
  end process comparator_behavior;

end architecture ideal;
```

The reference voltage is a constant value. How could this be changed to an extra input for reference source?

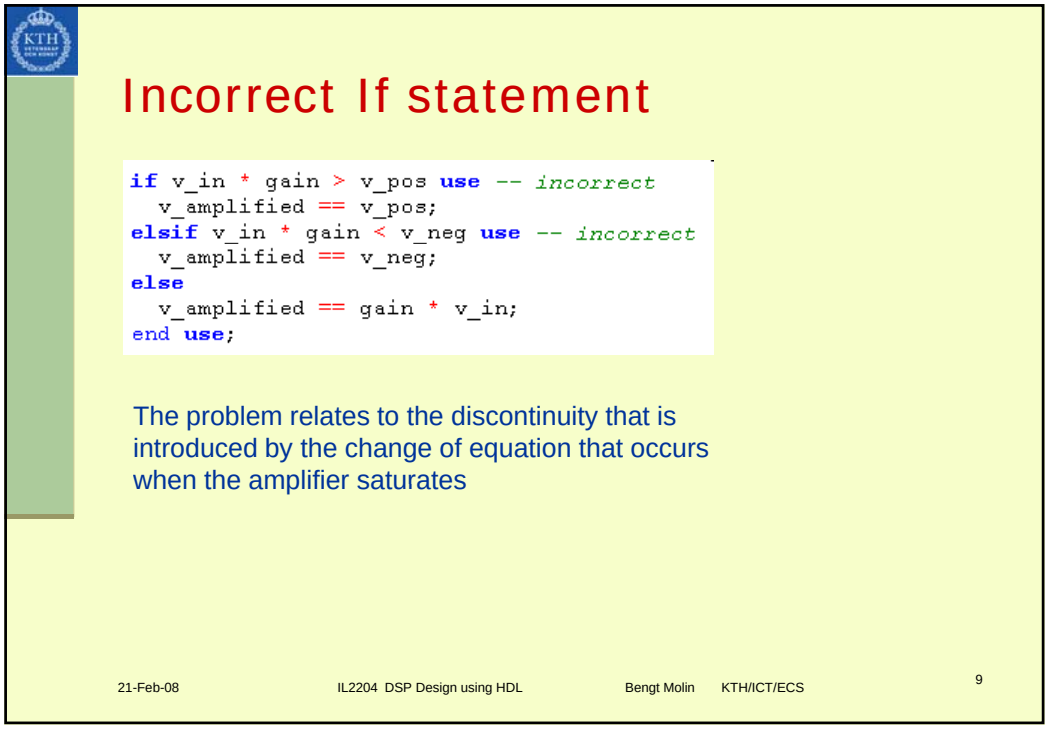
Simultaneous statements

Simple one

```
terminal p, m : electrical;
quantity v across i through p to m;


v == i * R;
```


Simultaneous If statement

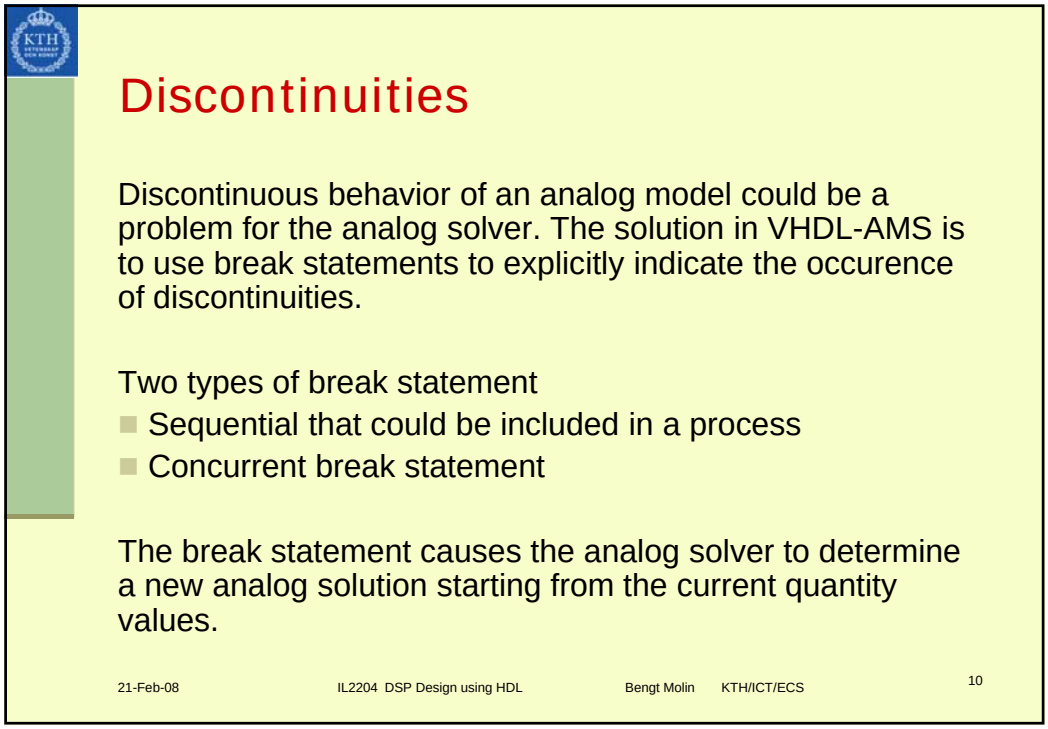
```
library ieee_proposed;  use ieee_proposed.electrical_systems.all;

entity opamp is
  port ( terminal positive_supply, negative_supply : electrical;
         terminal plus_in, minus_in, output : electrical );
end entity opamp;

architecture saturating of opamp is
  constant gain : real := 50.0;
  quantity v_pos across positive_supply;
  quantity v_neg across negative_supply;
  quantity v_in across plus_in to minus_in;
  quantity v_out across i_out through output;
  quantity v_amplified : voltage;

begin
  if v_in'above(v_pos / gain) use
    v_amplified == v_pos;
  elsif not v_in'above(v_neg / gain) use
    v_amplified == v_neg;
  else
    v_amplified == gain * v_in;
  end use;

  break on v_in'above(v_pos/gain), v_in'above(v_neg/gain);

  v_out == v_amplified'slew(1.0e6,-1.0e6);

end architecture saturating;
```



Incorrect If statement

```
if v_in * gain > v_pos use -- incorrect
  v_amplified == v_pos;
elsif v_in * gain < v_neg use -- incorrect
  v_amplified == v_neg;
else
  v_amplified == gain * v_in;
end use;
```

The problem relates to the discontinuity that is introduced by the change of equation that occurs when the amplifier saturates



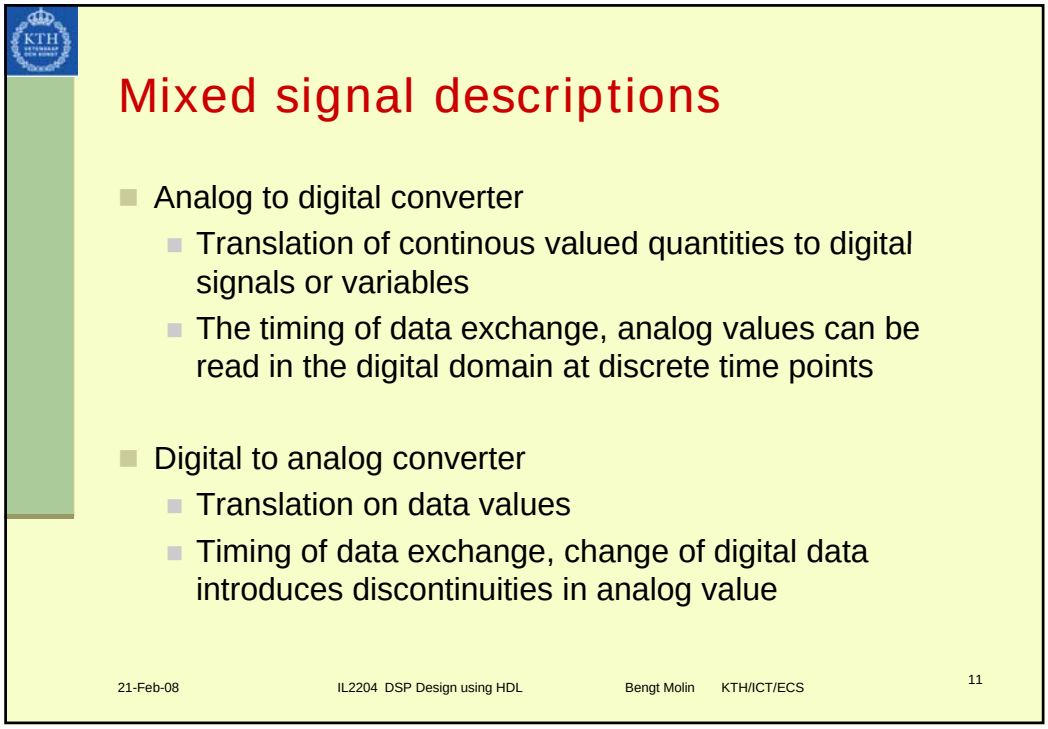
Discontinuities

Discontinuous behavior of an analog model could be a problem for the analog solver. The solution in VHDL-AMS is to use break statements to explicitly indicate the occurence of discontinuities.

Two types of break statement

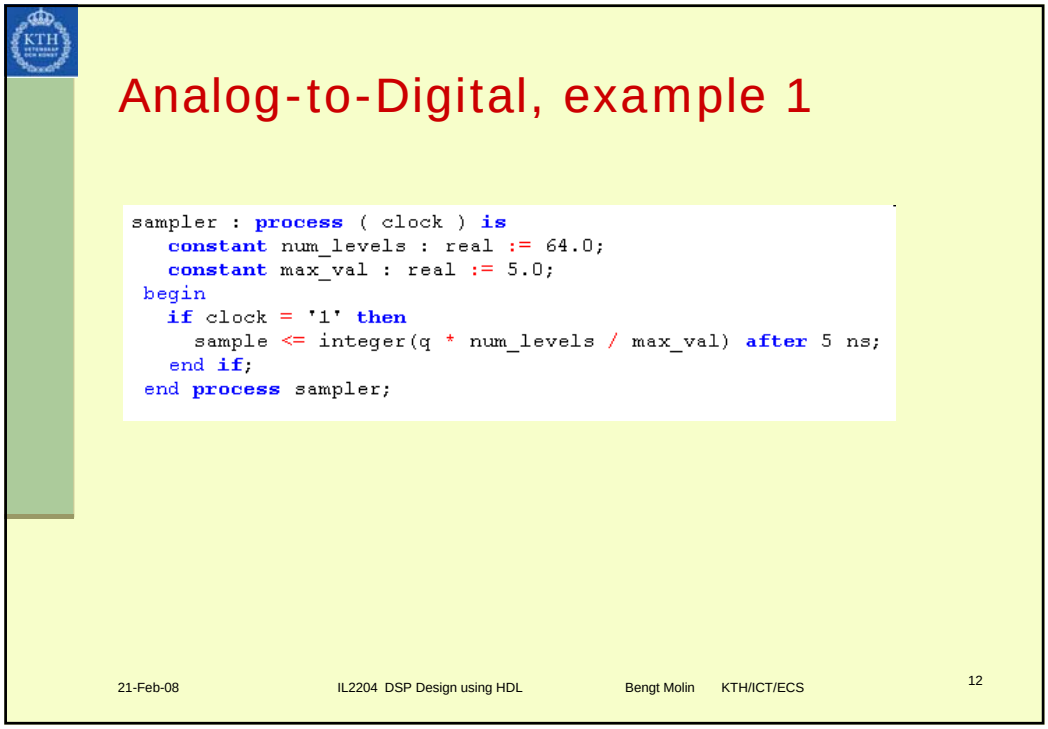
- Sequential that could be included in a process
- Concurrent break statement

The break statement causes the analog solver to determine a new analog solution starting from the current quantity values.



Mixed signal descriptions

- Analog to digital converter
  - Translation of continous valued quantities to digital signals or variables
  - The timing of data exchange, analog values can be read in the digital domain at discrete time points
- Digital to analog converter
  - Translation on data values
  - Timing of data exchange, change of digital data introduces discontinuities in analog value



Analog-to-Digital, example 1

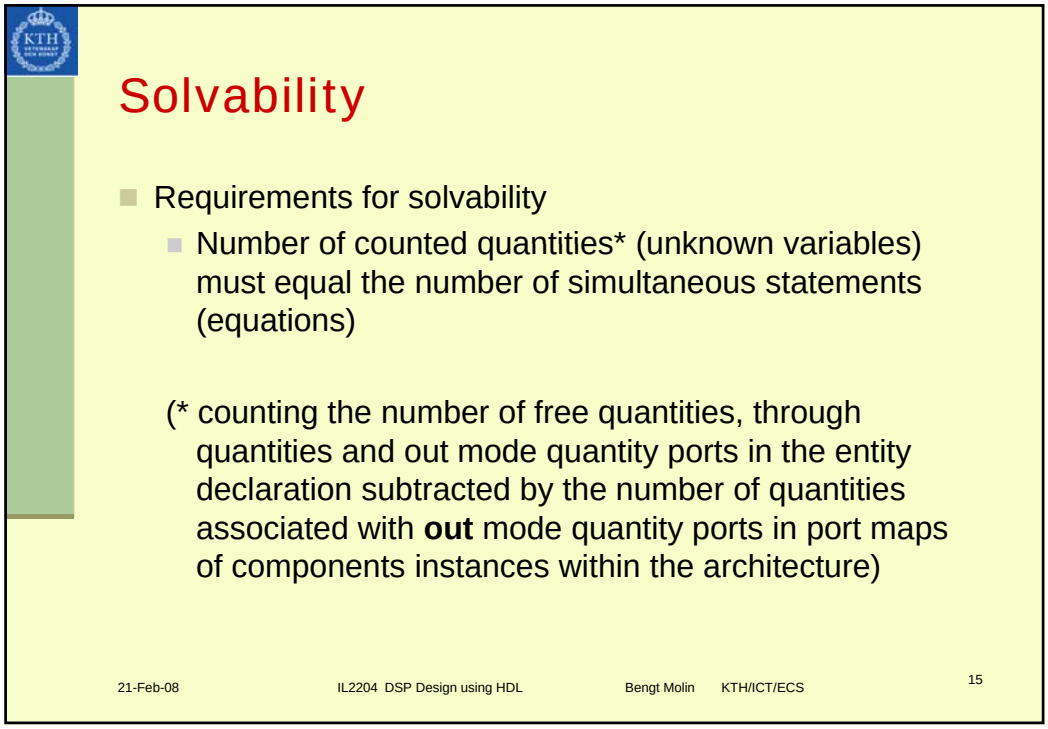
```
sampler : process ( clock ) is
   constant num_levels : real := 64.0;
   constant max_val : real := 5.0;
 begin
   if clock = '1' then
     sample <= integer(q * num_levels / max_val) after 5 ns;
   end if;
 end process sampler;
```

Analog-to-Digital, example 2

```
analog_to_std_logic : process (v_in'above(v_il), v_in'above(v_ih)) is
begin
  if not v_in'above(v_il) then
    data <= '0';
  elsif v_in'above(v_ih) then
    data <= '1';
  else
    data <= 'X';
  end if;
end process analog_to_std_logic;
```


Digital-to-Analog, example

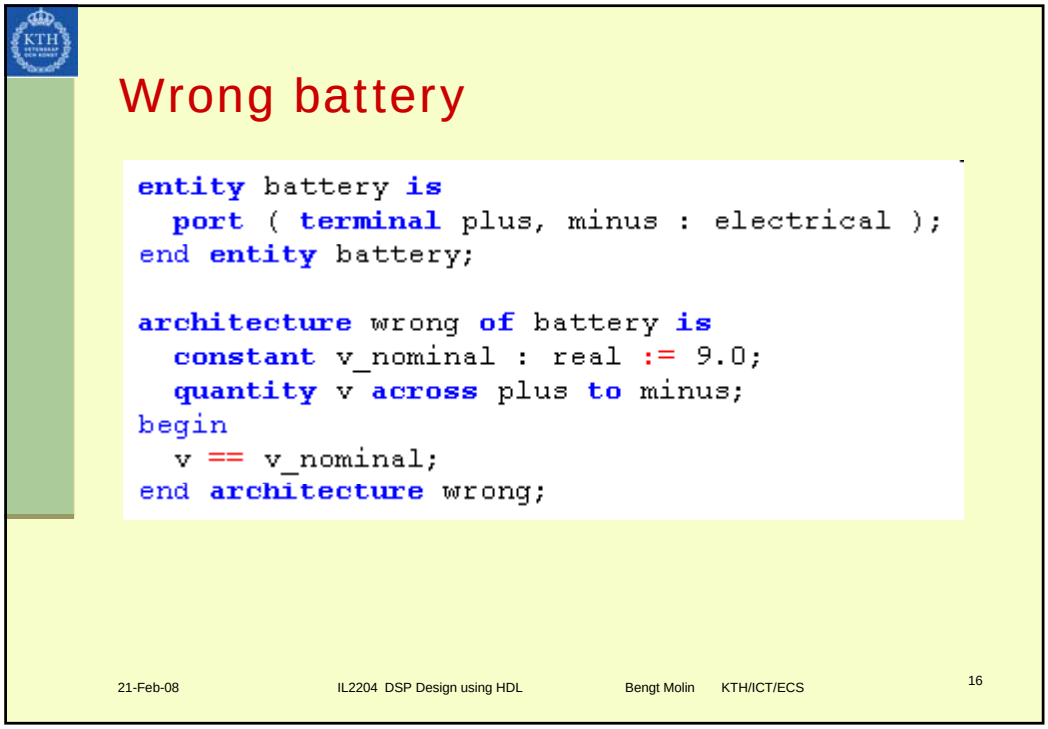
```
library ieee;  use ieee.std_logic_1164.all;
library ieee_proposed;  use ieee_proposed.electrical_systems.all;

entity dac_12_bit is
  port ( signal  bus_in : in std_ulogic_vector (11 downto 0);
         terminal analog_out :  electrical );
end entity dac_12_bit;

architecture behavioral of dac_12_bit is

  constant v_max : real := 3.3;
  signal s_out : real := 0.0;
  quantity v_out across i_out through analog_out to electrical_ref;

begin

  convert : process ( bus_in ) is
    variable sum : natural;
  begin
    sum  := 0;
    for i in bus_in'range loop
      sum := sum * 2 + boolean'pos( bus_in(i) = '1' or bus_in(i) = 'H' );
    end loop;
    s_out <= v_max * real(sum) / real(2**12 - 1);
  end process convert;

  v_out == s_out'ramp(1.0E-6);

end architecture behavioral;
```



Solvability

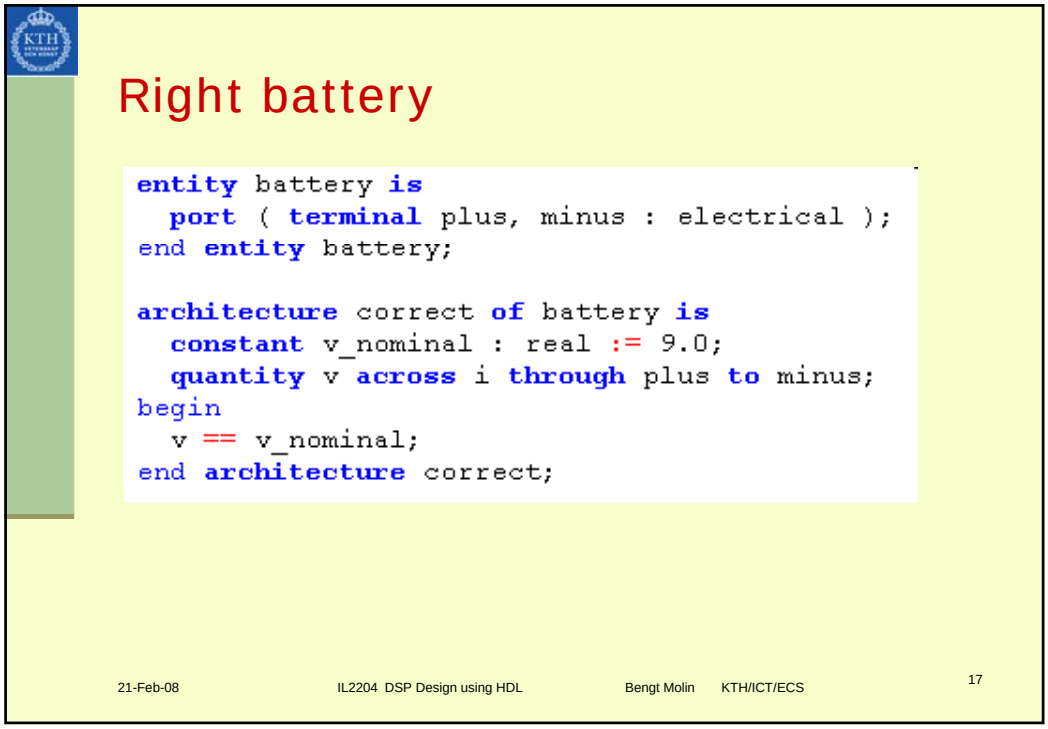
- Requirements for solvability
  - Number of counted quantities* (unknown variables) must equal the number of simultaneous statements (equations)

(* counting the number of free quantities, through quantities and out mode quantity ports in the entity declaration subtracted by the number of quantities associated with **out** mode quantity ports in port maps of components instances within the architecture)



Wrong battery

```
entity battery is
  port ( terminal plus, minus : electrical );
end entity battery;

architecture wrong of battery is
  constant v_nominal : real := 9.0;
  quantity v across plus to minus;
begin
  v == v_nominal;
end architecture wrong;
```


Right battery
entity battery is
port (terminal plus, minus : electrical);
end entity battery;
architecture correct of battery is
constant v_nominal : real := 9.0;
quantity v across i through plus to minus;
begin
v == v_nominal;
end architecture correct;
21-Feb-08
IL2204 DSP Design using HDL
Bengt Molin
KTH/ICT/ECS
17