Verilog / AMS

Enrico Fraccaroli
Franco Fummi

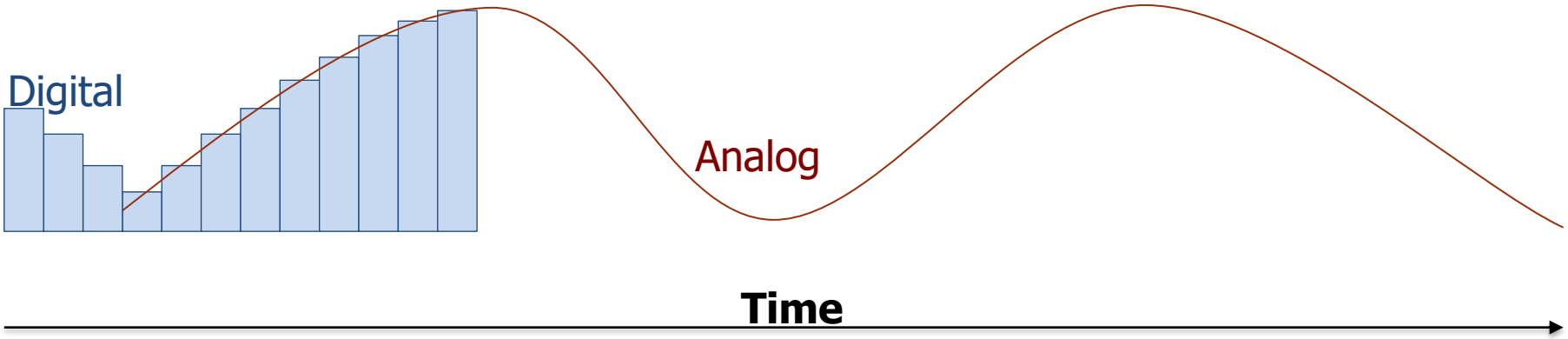
UNIVERSITÀ
di VERONA
Dipartimento
di INFORMATICA

Version 1.0

Contents

- Motivations
- Code structure
- Circuit structure
- Branch contributions
- Disciplines
- Natures
- Functions

Analog vs Digital (1/2)

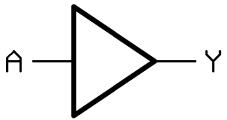


Analog vs Digital (2/2)

Truth Table

A	Y
0	1
1	0

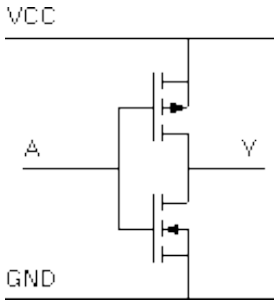
Digital Circuit



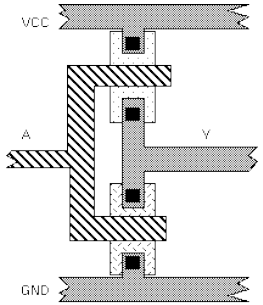
Analog Behavior

$$V(Y) = VCC - V(A)$$

Analog Circuit



Polygon Layout



Abstraction Level

Hardware Description Languages

SystemVerilog includes Verilog-AMS → Verilog/VHDL, Verilog-A

Verilog-AMS

Abstraction Level

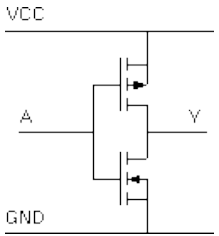
Verilog-A

Extension of Verilog to model analog parts

V(out) = VCC – V(in)

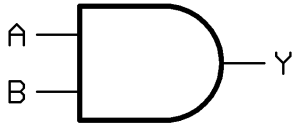
Behavioral

SPICE → Simulation kernel for transistors

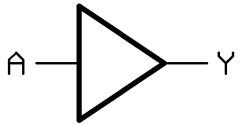


Analog

Verilog/VHDL



Digital



Circuit Structure

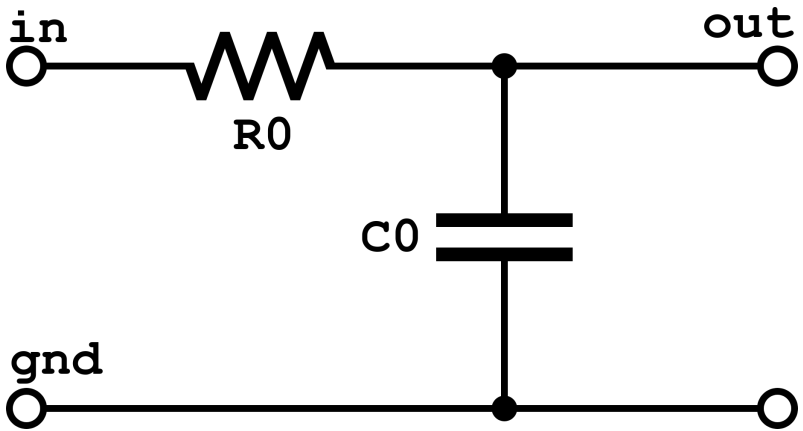
```
`include "disciplines.vams"

module rc_block(in, out, gnd);
  input in, gnd;
  output out;
  electrical in, out, gnd;
  parameter real R0 = 100.0;
  parameter real C0 = 0.0001;
  analog begin
    I(in, out) <+ V(in, out) / R0;
    I(out, gnd) <+ ddt(V(out, gnd)) * C0;
  end
endmodule
```

this means 'in', 'out', 'gnd' must follow Kirkhoff's laws

Conservative ⇐ Kirkhoff's laws

I don't have pre-existing modules for base components



Code Structure

```
`include "disciplines.vams"

module rc_block(in, out, gnd);
  input in, gnd;
  output out;
  electrical in, out, gnd;
  parameter real R0 = 100.0;
  parameter real C0 = 0.0001;
  analog begin
    I(in, out) <+ V(in, out) / R0;
    I(out, gnd) <+ ddt(V(out, gnd)) * C0;
  end
endmodule
```

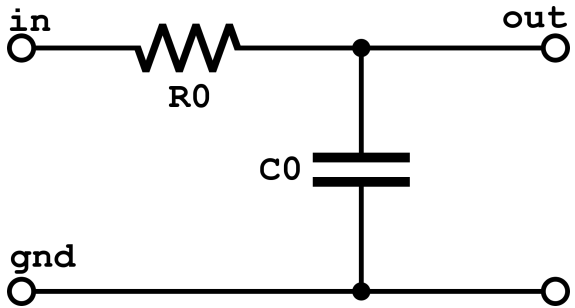
Interface

Ports

Nodes Type

Parameters

Behavior



Branch Contributions

The line:

```
I(in, out) <+ V(in, out) / R0;
```

defines the relationship between the module ports **out** and **in**. This is known as a **branch contribution** and is one of the most important Verilog-A concepts

Both **`V()` and `I()` functions** in the above are known as an **access function**

- `V()`: provides the **potential difference between the two nodes**
- `I()`: provides the **current flowing between the two nodes**

Letter V stands for Voltage, while `I` is the Current (letter I comes from the French, *intensité de courant*)

Disciplines (1/2)

The line:

```
electrical in, out, gnd;
```

defines the **discipline** for the module ports and ground node in this case '**electrical**'

Verilog-AMS supports other disciplines such as **thermal**, **mechanical** and **rotational** allowing **simulation** of **physical processes** other than electrical and electronic

The definitions of these other disciplines are defined in the disciplines.vams file which is included using the line:

```
`include "disciplines.vams"
```

Disciplines (2/2)

The list of disciplines supported by Verilog-AMS:

Name	Potential	Flow	Domain
logic	–	–	discrete
electrical	Voltage	Current	continuous
thermal	Temperature	Power	continuous
kinematic	Position	Force	continuous
rotational	Angle	Angular_Force	continuous

Voltage and Current have an equivalent in all the other disciplines Eg: in thermal, Temperature can be represented as Voltage

Natures

Potential and **Flow** of disciplines are selected from a table of **Natures**

Name	Units	Access
Voltage	V	V
Current	A	I
Charge	coul	Q
Temperature	K	Temp
Position	m	Pos
Velocity	m/s	Vel
Acceleration	m/s^2	Acc
Impulse	m/s^3	Imp
Force	N	F
Angle	rads	Theta
Angular_Force	N-m	Tau

Functions

The line:

```
I(out, gnd) <+ ddt(V(out, gnd)) * C0;
```

contains a function called **ddt**. An **analog operator** that performs the time derivative of the passed argument. There are many other analog operators in Verilog-AMS

Operator	Description
ddt(operand, [abstol|nature])	Time derivative
idt(operand, [ic], [assert], [abstol|nature])	Time integral
transition(operand, delay, trise, [tfall])	Transition
slew(operand, [rising_sr], [falling_sr])	Slew
absdelay(operand, delay, [max_delay])	Delay
laplace_zp(operand, [zeta], [rho], [epsilon])	Laplace, zero-pole form
laplace_nd(operand, [n], [d], [epsilon])	Laplace, numerator-denominator form
last_crossing(operand, [direction])	Last crossing
limexp(operand)	Limited exponential

These functions allow us to join and mix analog and digital simulations

⇓

Move values from analog part to digital, and viceversa