Verilog-A Language

By

William Vides

Edited by Dr. George Engel

Topics to be Covered

- Background information
- Analog System Description and Simulation
- Types of Analog systems
- Signals in Analog systems
- Analog System simulation
- Analog Model Properties
- Analog Operators

Background Information

- Fundamental differences between digital and analog design.
- Current level of abstractions achieved by Spice and Verilog HDL
- Verilog -A as an extension of Spice

Difference between Digital and Analog Design

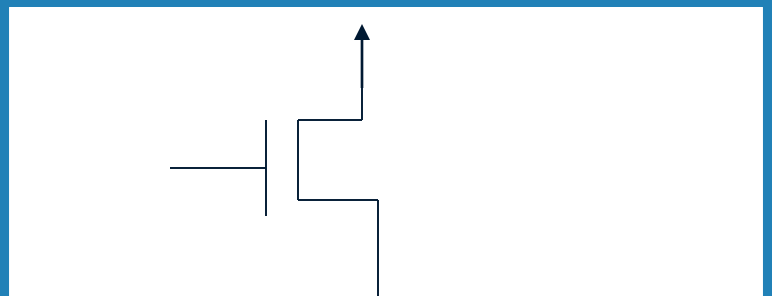
```
Always @ (enable)  begin
valid = 1'b0;
// do write cycle
addr_lines = addr;
data_lines = data;
@ (negedge clk) begin
    valid = 1'b1;
  end
```

Top Down
Refined from HDL

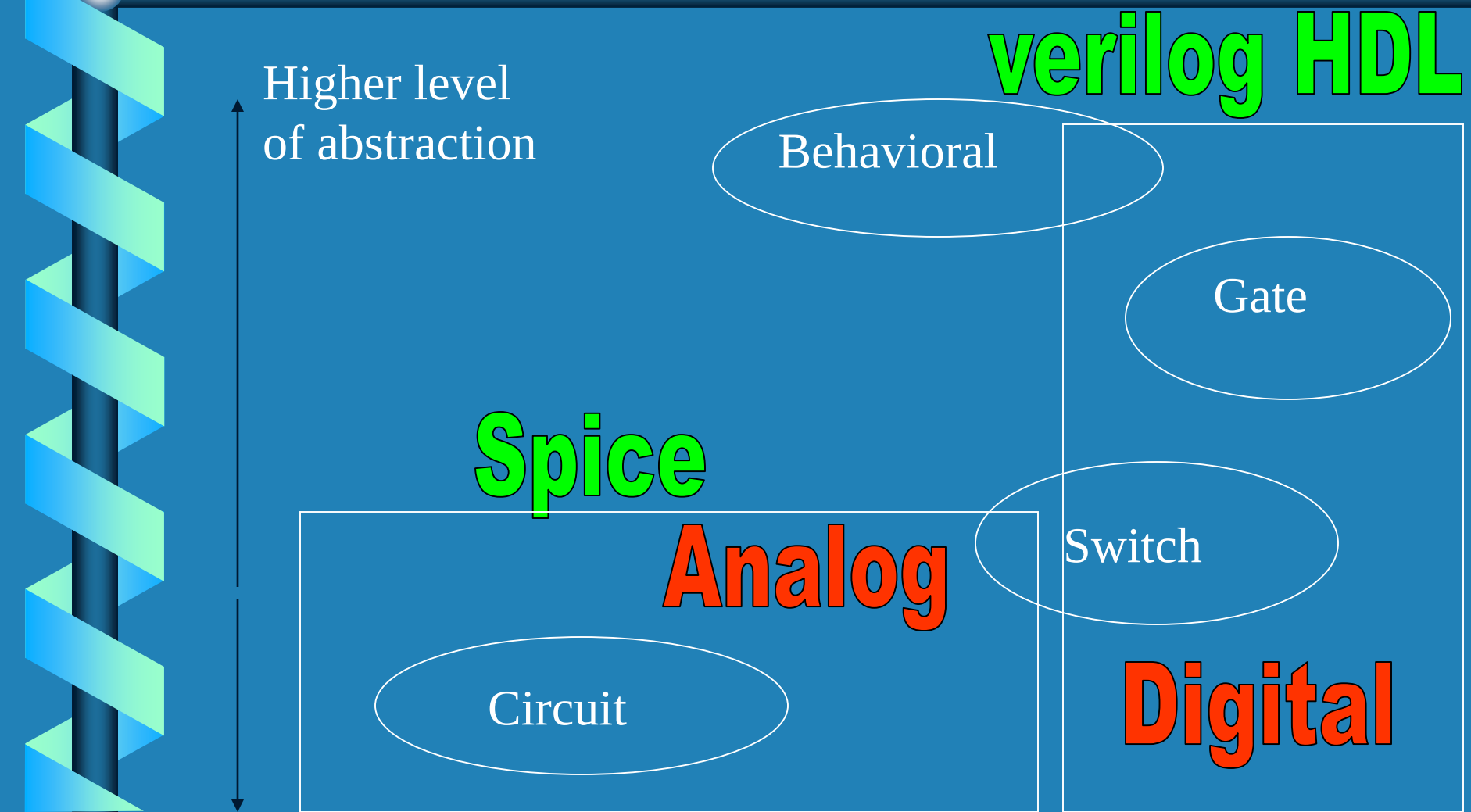
Digital

Analog

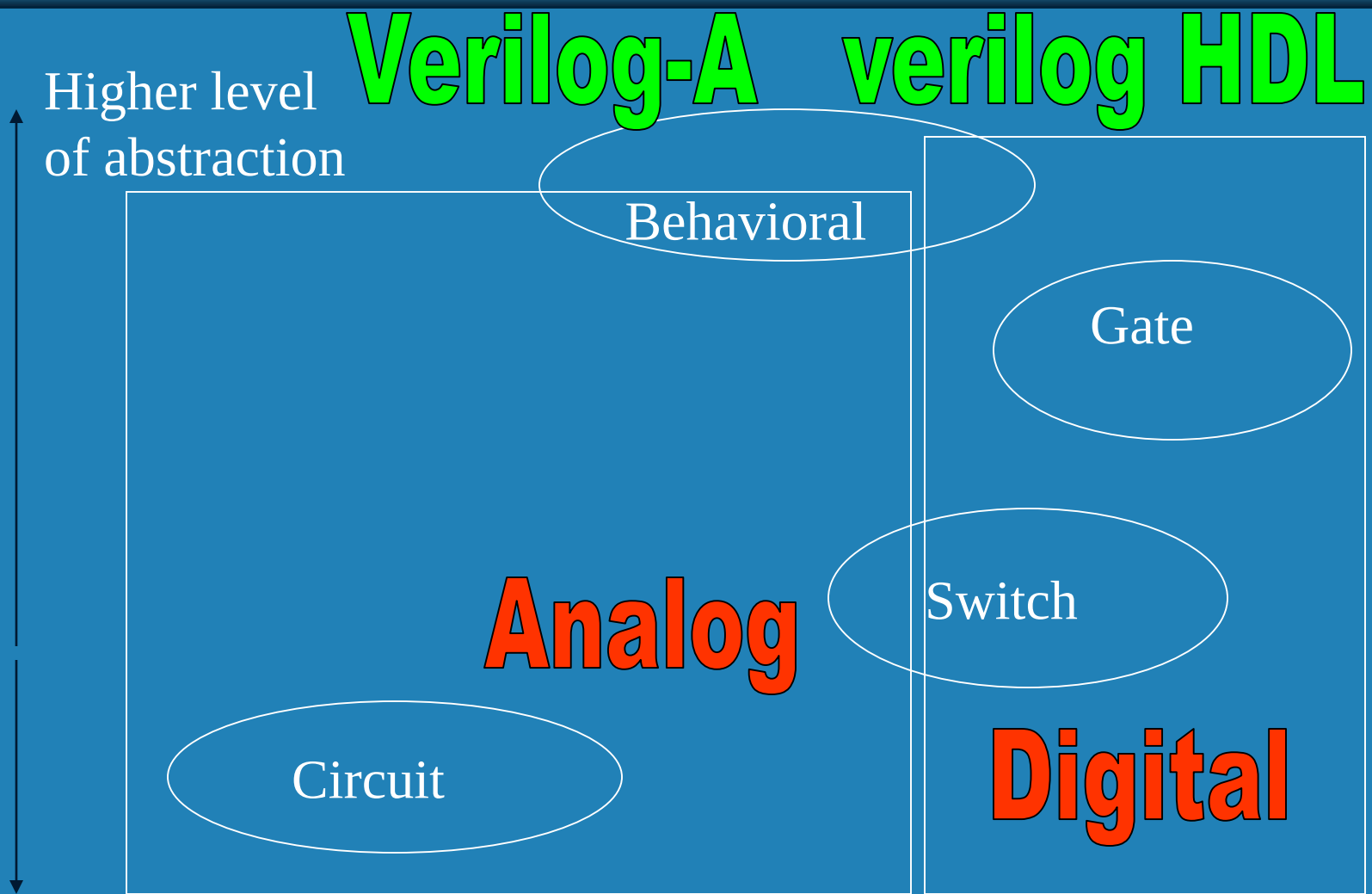
Bottom-Up
Transistor level



Current Levels of Abstraction Achieved by Spice and Verilog



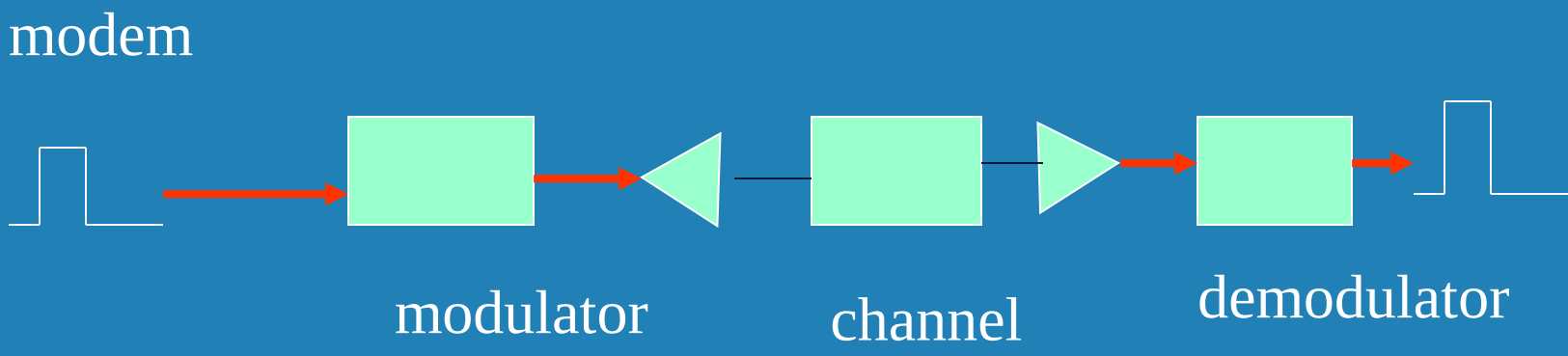
Verilog-A as an extension of Spice



Analog System Description and simulation

- Structural Description
  - a module is comprised of other child modules
- Behavioral Description
  - descriptions in a programmatic fashion with the Verilog-A language
  - The module is defined in terms of the values for each signal
- Mixed-level Descriptions
  - Combine both Structural and Behavioral Descriptions

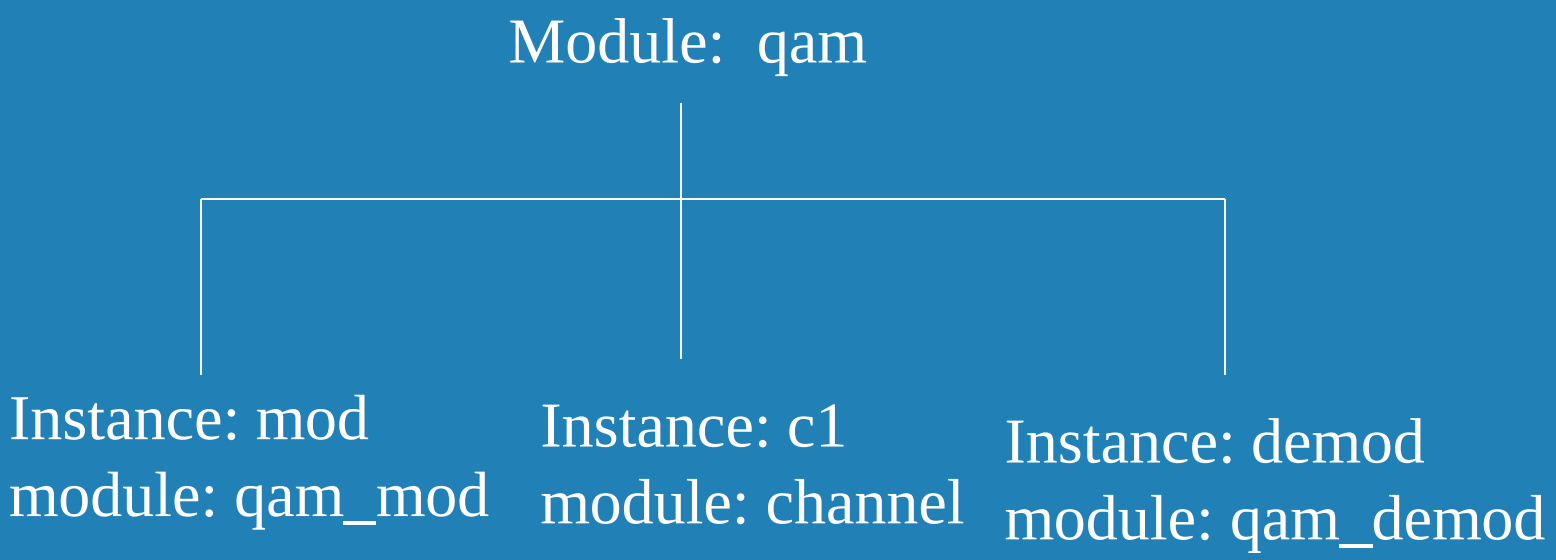
Modem Example



The modem system is made up of

1) the modulator
2) a channel
3) the demodulator

Structural Description hierarchy



Structural Description of the Modem System

```
// Verilog A definition of the modem System
`include "std.va"

module modem( dout, din)'
   inout  dout, din;
   electrical dout, din;
parameter real fc = 100.0e6;

electrical clk, cin, cout;
qam_mod #(.carrier_freq(fc)) mod (cin,din,clk);
channel c1 ( cout, cin);
qam_demod #(.carrier_freq(fc)) demod (dout,cout,clk);
endmodule
```

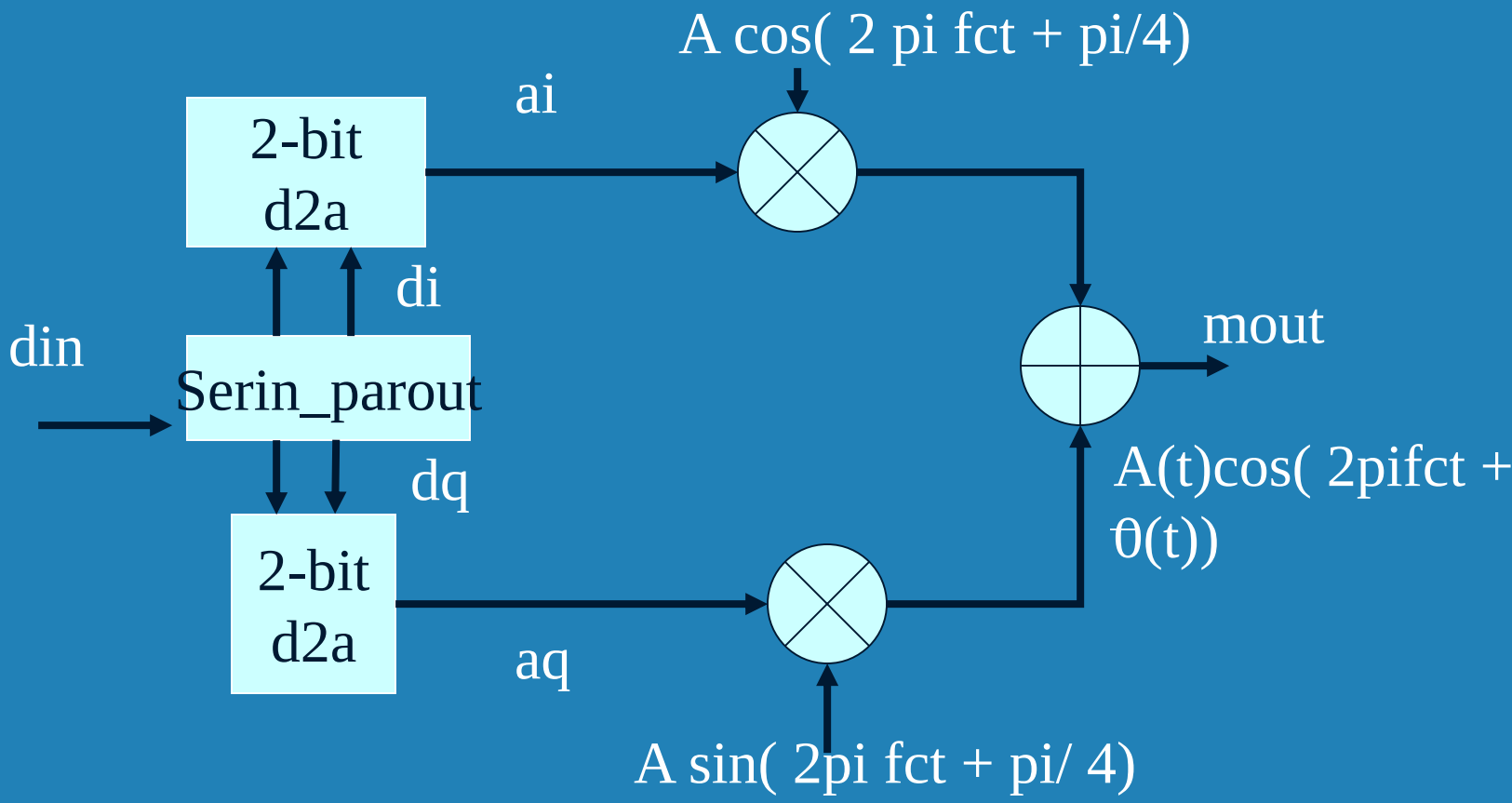
Structural Description

Type of module instance

Name of the instance created

Parameter name in child (qam_mod) module assigned as: carrier_freq = fc

16_QAM modem Example



Verilog A mixed Signal definition of 16-QAM modulator

```
`include "std.va"
`include "const.va"
module qam_mod( mout, din, clk);
  inout mout, din, clk;
  electrical mout, din, clk;
parameter real fc = 100.0e6;
  electrical di1,di2, dq1, dq2;
  electrical ai, aq;
    serin_parout sipo( di1,di2,dq1,dq2,din,clk);
   d2a d2ai(ai, di1,di2,clk);
  d2a d2aq(aq, dq1,dq2,clk);
 real phase;
```

Verilog A mixed Signal definition of 16-QAM modulator

```
analog  begin
  phase = 2.0 * `M_PI * fc* $realtime() + `M_PI_4;
V(mout) <+ 0.5 * (V(ai) * cos(phase) + V(aq) * sin (phase));
  end
endmodule
```

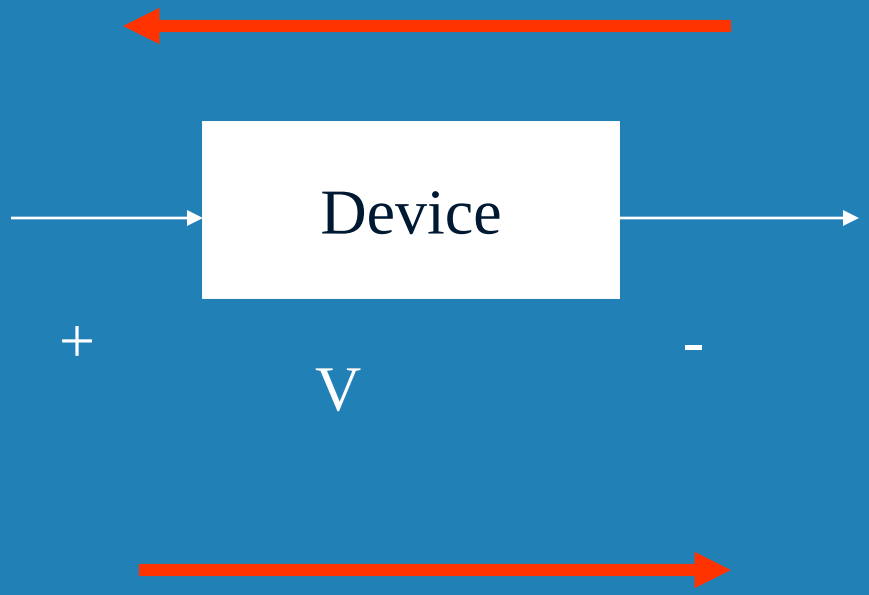
The behavioral definition of the QAM modulation is defined

The signals ai and aq are the outputs of the 2-bit D/A converters

Type of analog systems

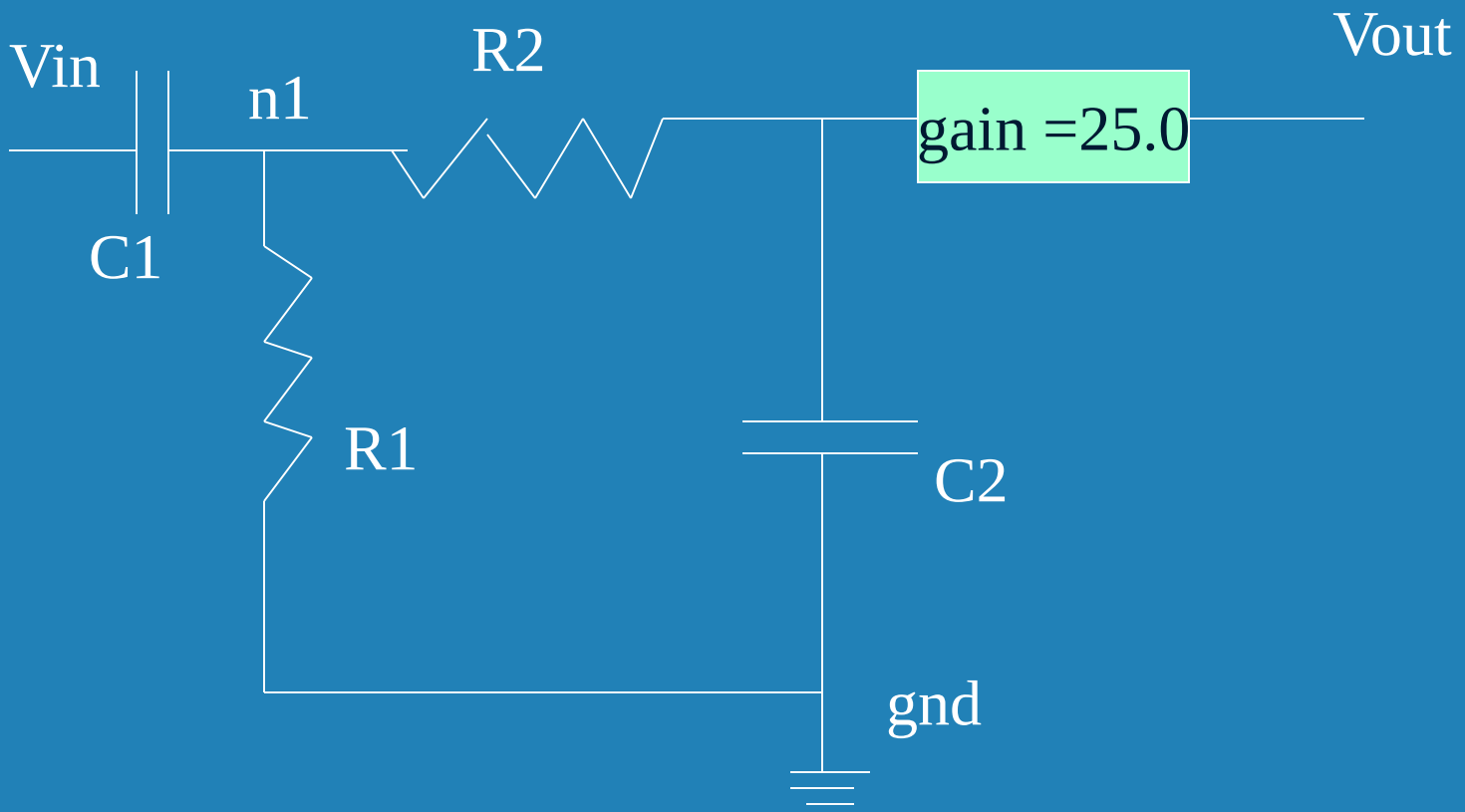
- Conservative Systems
  - use of Kirchoff's laws
  - Electrical Systems use KVL and KCL
  - Any conservative System use KPL and KFL
    - applied to branches
- Signal Flow Systems
  - only potential is associated with every node
  - unidirectional
  - notion of ports (input / output)

Conservative Systems



In a conservative system the charges or signals can enter a particular device in both ways.

Common Emitter amplifier with RC bandpass filter Example



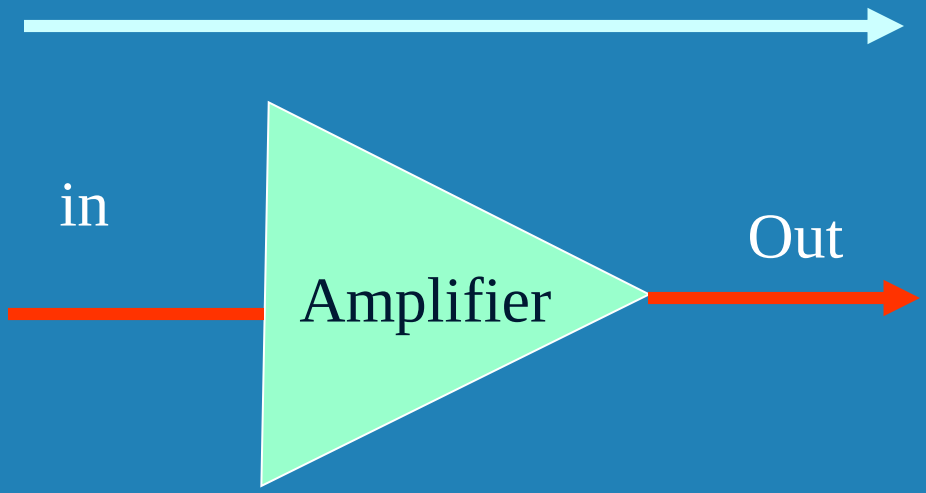
Verilog A behavioral description of ce-amp with RC bandpass filter

```
`include "std.va"
module mbce_amp_rcn ( in, out, gnd);
   inout in, out, gnd;
   electrical in, out, gnd;

parameter real gain = 1.0;
parameter real r1 = 4k;
parameter real c1 = 100n;
parameter real r2 = 100k;
parameter real c2 = 2.8p;

   electrical n1, n2;
```

Verilog A behavioral description of ce-amp with RC bandpass filter

```
analog  begin
        I( in, n1) <+ c1 * ddt( V(in,n1));
        V(n1, gnd) <+  r1 *  I(in,n1);
        I(n1,n2) <+ V(n1, n2) / r2;
        I(n2,gnd) <+ c2* ddt( V(n2, gnd));
        V(out, gnd) <+ V(n2,gnd) * (-gain);
 end

endmodule
```

Signal Flow Systems



In signal flow systems a signal can only enter a device in one way only.

What is Simulation?

- simulation is a process in which a system of nonlinear ordinary differential equations is solved
- this equations are not input directly , but derived from each of the models that are interconnected in the netlist

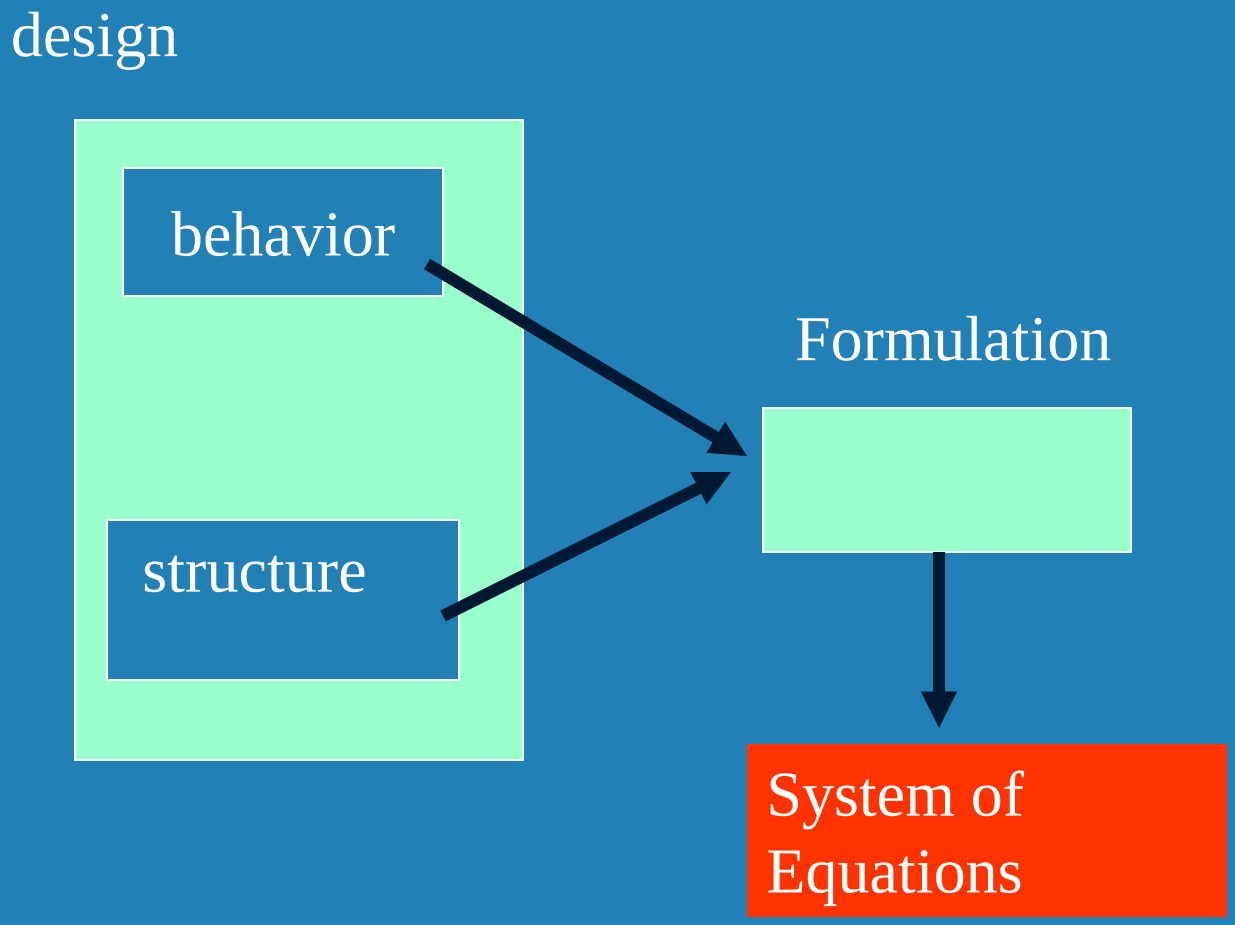
What it means to the user ?

- To the user a simulation is essentially a software version of an oscilloscope or logic analyzer.
- A simulation is a technique by which the user ask questions and receives answers from a program .
- The quality of the answers depends on the quality of the questions.

Analog System Simulation

- The Standard approach to analog circuit simulation involves
  - formulate the differential-algebraic equations for the circuit
  - applying implicit integration methods to the sequence of nonlinear algebraic equations
  - iterative methods such as Newton-Raphson to reduce to a set of linear equations
  - using sparse matrix techniques to solve the linear equations

Analog System Simulation



Analog Model Properties

- The Verilog-A language can be used to represent different types of behaviors these include
  - Linear
  - Nonlinear
  - Piecewise linear
  - Integro differential
  - Event-driven Analog

Analog Operators

- The Verilog-A language defines analog operators for
  - Time Derivative
  - Time Integral
  - Linear time delay
  - Discrete waveform filters
  - LaPlace Transform filters
  - Z-transform filters

Time Derivative Operator

- The ddt Operator computes the time derivative of its arguments
  - ddt (expression)
- In DC analysis the ddt operator returns a zero.
- Application of the ddt operator results in a zero at the origin.

Time Integral Operator

- The idt operator computes the time integral of its arguments
  - idt(expression, ic, reset)
- When specified with initial conditions the idt operator returns the value of the initial condition in DC.
- Without initial conditions , idt multiplies its argument by infinity in DC analysis.

Time Integral Operator

- The Optional argument RESET allows resetting of the integrator to the initial condition or IC value.
- Application of the idt operator results in a pole at the origin.

Delay Operator

- Delay operator implements a transport or linear time delay for continuous waveforms
  - delay (expression, dt)
- The parameter DT must be positive
- The effect of the delay operator in the time domain is to provide a direct time translation of the input

Transition Operator

- The transition operator smooths out piece-wise constant waveforms.
- The transition filter is used to imitate transitions and delays on discrete signals
  - transition (expression, dt, tr, tf)
- The input expression to the transition operator must be defined in terms of discrete states.

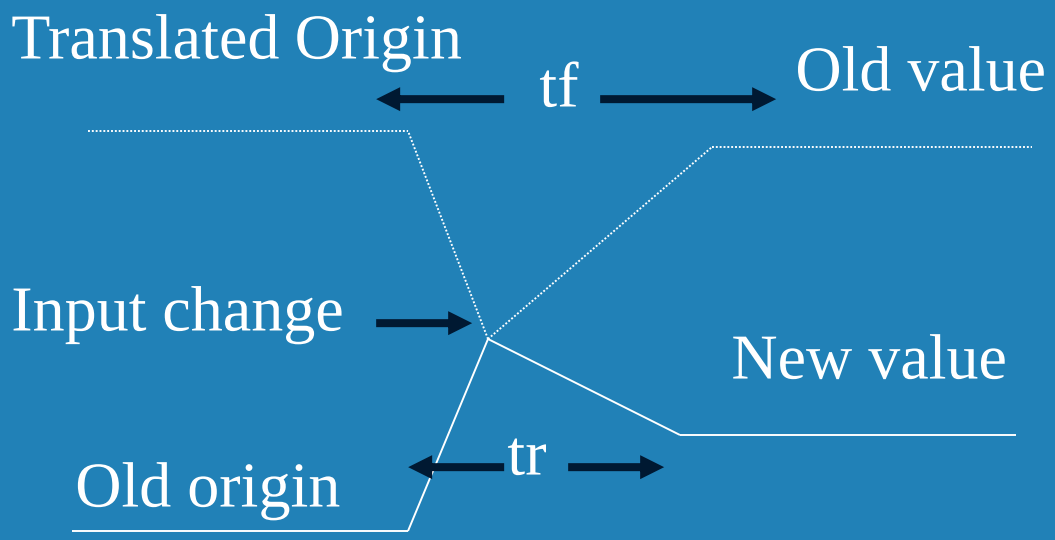
Transition Operator

- The parameters dt, tr, tf are optional
  - tr - transition rise
  - tf - transition fall
  - dt - change in time
- if dt is not specified then it is taken to be zero
- if the value for tr is specified the simulator will use it for both the rise and fall times.

Transition Operator

- When rise and fall times are longer than the specified delay
  - if the new final value level is below the current value the transition Operator uses the old destination as the origin.
  - If the new destination is above the current level the first origin is retained

Transition Operator



A rising transition is interrupted near its midpoint, and the new destination level of the value is below the current value. For the new origin and destination. The transition computes the slope that completes the transition from origin in the specified transition time. It then uses the computed slope to transition from the current value to the new destination.

Slew Operator

- The slew operator bounds the slope of the waveform
- used to generate continuous signals from piece-wise continuous signals
  - slew (expression, mpsr, mnsr)
  - mpsr - maximum positive slew rate
  - mnsr - minimum negative slew rate

Slew Operator

- The Slew Operator forces all transitions of the input expression faster than mpsr to change at mpsr for positive transitions and limits negative transitions to mnsr
- mpsr must be greater than zero
- mnsr must be lower than zero
- if only one rate is specified, the absolute value will be used for both rates

Slew Operator

- If no rate is specified the slew operator passes the signal through unchanged.
- In DC analyses, the slew operator passes the value of the destination to its output
- In AC small-signal analyses the slew function has unity transfer function
  - except when slewing, in that case it has zero transmission through the slew operator

Laplace Transform Operators

- The Laplace transform operators implement lumped, continuous-time filters
  - laplace_zp(express, numerator, denominator)
  - lapace_zd(express, numerator,denominator)
  - lapace_np(express, numerator,denominator)
  - lapace_nd(express, numerator,denominator)
    - H(s) = N(s)/D(s)

Laplace Transform Operator

- The laplace transform analog operator take vector arguments that specify the coefficients of the filter
- Laplace analog operators represent linear time-Invariant filters
- laplace_zp - the zeros and poles are specified as pairs of real numbers
  - specifying the real and imaginary components of each zero or pole

Laplace Transform Operator

- Laplace_nd - zeros and poles of the filter are specified as polynomial coefficients from lowest order term to highest

- Laplace_zd - zeros of the filter are specified as pairs of real numbers and the poles of the filter are specified as polynomial coefficients

Laplace Transform Operator

- Laplace_np - Zeros of the filter are specified as polynomial coefficients and the poles of the filter are specified as pairs of real numbers

Laplace Transform example

```
// Laplace analog operator example of Butterworth low-pass
// filter using laplace_nd
module  laplace_op(out , in);
    inout out, in;
    electrical out, in;

   analog
          V(out) <+ laplace_nd ( V(in), {1.0},
                        {1.0, 3.236, 5.236, 5.236, 3.236, 1.0});
endmodule
```

Taken from the equation

H(s) = 1/(s^5 + 3.236s^4 +5.236s^3 +5.236s^2 +3.236s +1)

Z-Transform Operators

- The Z-Transform operators implement linear discrete-time filters
  - zi_zp(expression, numerator,denominator,T ,trf ,t0)
  - zi_zd(expression, numerator,denominator,T ,trf ,t0)
  - zi_np(expression, numerator,denominator,T ,trf ,t0)
  - zi_nd(expression, numerator,denominator,T ,trf ,t0)

Z-Transform Operators

- H(z) = N(z)/ D(z)

- the Z-transform analog operator take vector arguments that specify the coefficients of the filter.
- All Z-transform share the arguments T, trf, and t0
  - T -specifies the period of the filter
    - mandatory and must be positive

Z-Transform Operators

- Trf - specifies the optional transition time and must be positive
  - if trf is zero, then the output is abruptly discontinuous
  - a Z-transform filter with zero transition time assigned directly to a source branch can generate discontinuities
- t0 - specifies the time of the first transition and is optional
  - if t0 is not given, the transition occurs at t=0

Z-Transform Operators

- Zi_zp - zeros and poles of the filter are specified as pairs of real numbers
  - specifying the real and imaginary components of each zero or pole
- zi_nd- zeros and poles of the filters are specified as polynomial coefficients
  - from the lowest order term to the highest

Z-Transform Operators

- Zi_zd - zeros of the filter are specified as pairs of real numbers and the poles of the filter are specified as polynomial coefficients
- zi_np- zeros of the filters are specified as polynomial coefficients and the poles of the filter are specified as pairs of real numbers