

# Verilog-A in SPICE

MOS-AK December 2014, Berkeley, CA.

Patrick O'Halloran, Tiburon Design Automation

# Verilog-A

Verilog-A has become the most commonly used analog HDL in SPICE; well defined and easy to use.

Defined by the Accellera LRM as the analog subset of Verilog-AMS

LRM 1.0(1996) -> 2.4(2013). Most implementations support 2.3 (2009)

Find the standard here

<http://www.accellera.org/downloads/standards/v-ams>

# Applications in SPICE

- General analog behavioral modelling
- Custom models in a PDK; sometimes wrapper existing compact models.
- Compact models; most are developed in VA; definition delivered to CMC in VA.
- Usually bsources are supported via Verilog-A
- The analog side of AMS; used in every connect module.

# Just include and instantiate

Create the Verilog-A code, save in an file typically with a .va extension, i.e. `ideal_tx.va`

Include in the SPICE netlist and instantiate

```
.hdl ideal_tx.va  
...  
x1 1 0 2 0 ideal_tx delay=1ns  
...
```

Run SPICE, your simulator will compile, elaborate and simulate with this new model

# A simple behavioral model

```
module ideal_tx(p1,n1,p2,n2);  
    electrical p1,n1,p2,n2;  
    parameter real td = 1.0e-9 from (0:inf);  
    parameter real Zo = 50 from [0:inf);  
analog begin  
        V(p2,n2) <+ Zo*I(<p2>) + absdelay(V(p1,n1), td)+Zo*absdelay(I(<p1>), td);  
        V(p1,n1) <+ Zo*I(<p1>) + absdelay(V(p2,n2), td)+Zo*absdelay(I(<p2>), td);  
end  
endmodule
```

# A SPICE macro model

## SPICE:

```
.subckt step_filter p1 p2
    t1 p1 0 int 0 z0=10 td=1e-9
    t2 int 0 p2 0 z0=90 td=1e-9
.ends
```

## Equivalent in Verilog-A:

```
module step_filter(p1, p2);
    electrical p1,p2;
    electrical gnd;
    ground gnd;
    tranline #(.z0(10), .td(1e-9)) t1(p1, gnd, int, gnd);
    tranline #(.z0(90), .td(1e-9)) t2(int, gnd, p2, gnd);
endmodule
```

# Structure and behavior

```
module step_filter(p1, p2);  
electrical p1,p2;  
electrical gnd;  
ground gnd;  
    tranzline #(.z0(10), .td(1e-9)) t1(p1, gnd, int, gnd);  
    tranzline #(.z0(90), .td(1e-9)) t2(int, gnd, p2, gnd);  
analog  
    $strobe ( "V(p1) : " , V(p1), "V(p2) : ", V(p2) );  
endmodule
```

# SPICE and the LRM

LRM describes Verilog-A mostly in the context of a Verilog-AMS simulator

But most VA usage in practice is from traditional netlist based SPICE

Most common usage involves non-hierarchical modules with no references to or from any other instance in the design.

These types of devices may be elaborated much like standard SPICE built-in devices. The traditional SPICE flattener needs no major modifications.



# Elaboration time issues for simple VA devices

- Module case, parameter case
- Parameter arrays (multidimensional), strings
- Buses
- Treatment of names with special characters
- Interfacing to the SPICE model card
- Multiplicity ( m,M, \$mfactor )

# More complex elaboration

- SPICE elaboration involves expanding relatively simple subcircuit definitions
- VA has a lot more flexibility in how to define and configure the design
- When we think of VA devices in a netlist as being part of a complete design then elaboration becomes more complex


# Moving from a subckt -> VA is simple

## Typical SPICE subcircuit instantiation line:

```
.subckt step_filter p1 p2
  t1 p1 int z0=10 td=1e-9
  t2 int p2 z0=90 td=1e-9
.ends
```

## Equivalent in Verilog-A:

```
module step_filter(p1, p2);
  electrical p1,p2;
  electrical gnd;
  ground gnd;
  tranline #(.z0(10), .td(1e-9)) t1(p1, gnd, int, gnd);
  tranline #(.z0(90), .td(1e-9)) t2(int, gnd, p2, gnd);
endmodule
```



# Verilog-A as just a better subcircuit

- Supports connect by name; less error prone, documents the intent and any port may be remain disconnected
- Support for buses, not just bus information encoded in node names
- Optional parameter range checking; parameters verified as valid during elaboration.

# With additional structural elaboration features, some examples:

- OOMRs, both among VA instances and to SPICE parameters and state variables
- Structural generate, conditionals and loops in structural code
- Verilog design configuration and SPICE hierarchical cell resolution.

# Strobe using an OOMR

## SPICE netlist:

```
.hdl oomr.va  
vs n1 0 1  
r1 n1 0 1  
x1 oomr  
.dc vs 0 10 1
```

## Verilog-A:

```
module oomr();  
analog  
    $strobe ( "V(n1) : ", v($root.n1) ); // 2.4  
endmodule
```

# RC line using generate and builtins

```
module rcline (n1, n2);  
...  
genvar i;  
  generate  
    for (i=0; i <N; i=i+1) begin  
      resistor #(.r(Rsec)) R(n[i], n[i+1]);  
      capacitor #(.c(Csec)) C(n[i+1], gnd);  
    end  
  endgenerate  
...  
endmodule
```

Elaboration result is same as if we had written a hard coded SPICE subcircuit. Core circuit simulator sees the same set of elements. All complexity is in the elaborator.

# Bus termination using iterated instance

## Top in SPICE:

```
.hdl term_bus.va  
x1 a<1> a<2> a<3> a<4> a<5> src  
x2 a<1> a<2> a<3> a<4> a<5> load
```

## Top in VA:

```
module top();  
    electrical [0:7] abus;  
    src #(.size(8)) s1(abus);  
    load #(.size(8)) l1(abus);  
endmodule
```

```
module src(abus);  
    parameter size=5;  
    electrical [1:size] abus;  
    ground gnd;  
    vsource #(.dc(1.0)) v1 [1:size] ( abus, gnd );  
endmodule  
  
module load(abus);  
    parameter size=5;  
    electrical [1:size] abus;  
    ground gnd;  
    resistor #(.r(1e3)) r1 [1:size] ( abus, gnd );  
endmodule
```



# VA Compact Models

Must always support full SPICE model feature set.

Always faster evaluation speed required in the NR loop

Always lower memory footprint required

More compiler diagnostics required to help the developer build a more efficient model

Very large instance count so though the representation is in VA these modules really act as SPICE primitives.

# Verilog-A LRM future releases

The Verilog-AMS committee are currently working on the System Verilog-AMS standard; initial white paper is expected in Spring 2015.

Verilog-A should remain a well defined subset but benefit from additional SV features

New releases should remain backward compatible