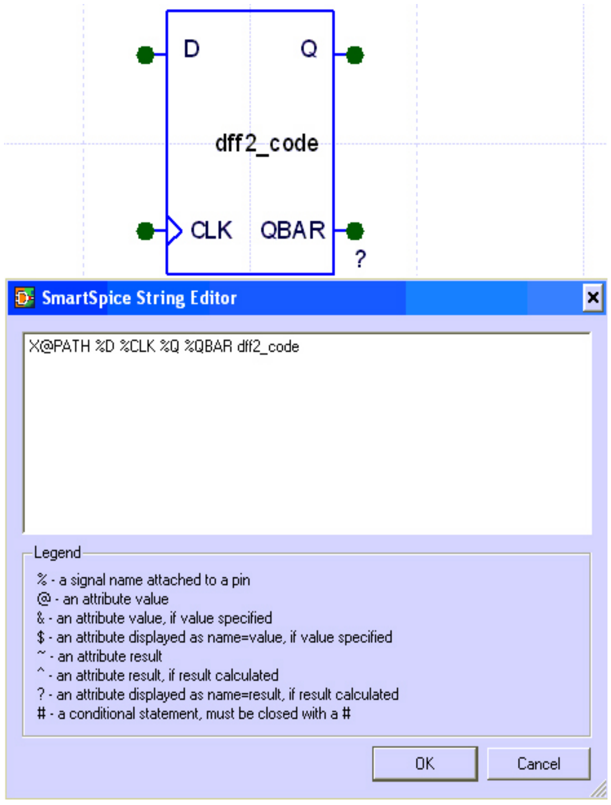
New Syntax for running Verilog-A Models in Gateway/SmartSpice

Including Verilog-A models in a simulation saves time and with a recent change in SmartSpice, it is even more convenient to add them to a circuit. Now Verilog-A models can be added just by calling the instance a subcircuit and adding a .VERILOG statement to include the Verilog-A module.

Verilog-A can be used to create exact circuits at the transistor level, but to prove that a circuit works, Behavioral models can be used in place of circuits with active devices. This allows large circuits to run in minutes instead of days. Even a simple circuit like the D Flip-Flop example (Figure 1) runs about 10 times faster using Verilog-A instead of simple level 3 transistors. For large complex circuits with high level models, the difference can easily be a hundred times faster using Verilog-A Behavioral models compared to using SPICE models



```
X@PATH %D %CLK %Q %QBAR dff2_code
```

Legend
% - a signal name attached to a pin
@ - an attribute value
& - an attribute value, if value specified
$ - an attribute displayed as name=value, if value specified
~ - an attribute result
^ - an attribute result, if result calculated
? - an attribute displayed as name=result, if result calculated
- a conditional statement, must be closed with a

Figure 1. SmartSpice String.

To add a Verilog-A device to Gateway, create a symbol and add the subcircuit call to the SmartSpice String Editor as in Example 1. By starting the string with an X, the device is designated as a subcircuit. The pins are listed in the same order as they are listed in the Verilog-A module, and the module name is listed. In the control card, add a .VERILOG call to the Verilog-A module. The Verilog-A module does not need to be in the same directory as the schematic, but the whole path must be included in the .VERILOG call if the module is not in the same directory. See the Verilog-A netlist for an example.

Figure 2 shows that the two devices have similar output, but closer inspection of the high-to-low transition in Figure 3 shows that the Verilog-A model is just a Behavioral model, not an exact match at the transistor level. However, to prove a concept or to replace a block where the transition is more important than the waveform, the results are sufficiently close.

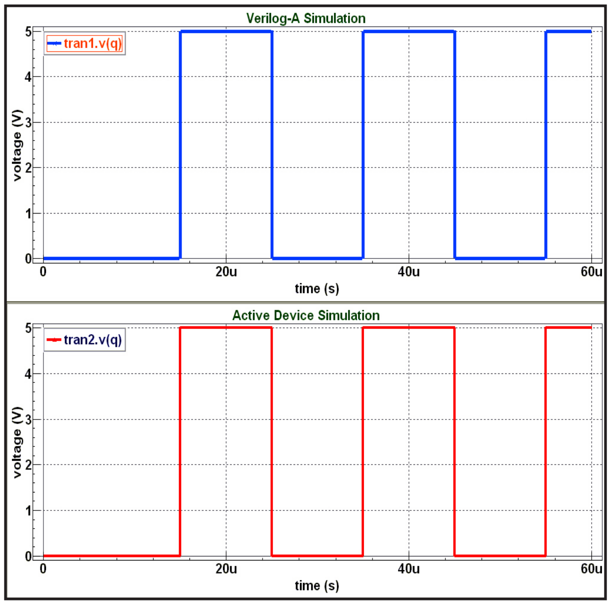


Figure 2. Results from the two simulations #1: Verilog-A vs. SPICE

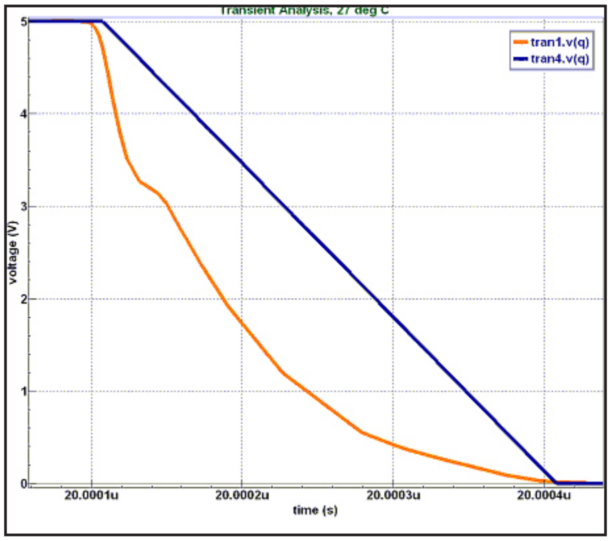


Figure 3. Results from the two simulations #2: Verilog-A vs. SPICE

Verilog Nelist

```
*dff_sim
* Gateway 2.6.7.R Spice Netlist
Generator
* Simulation timestamp: 27-Sep-2007
10:57:26
**
* Schematic name: dff_sim
*
V2 clk GND PULSE(0 5 5u 0.1n 0.1n 5u
10u)
V3 VDD GND DC 5
V4 d GND PULSE(0 5 10u 0.1n 0.1n 10u
20u)

** Verilog-A instance
X1 d clk q qbar dff2_code
*
* Global Nodes Declarations
.GLOBAL  clk GND VDD
* End of the netlist
*
* Markers to save
.SAVE ALL(V) V(q)

.VERILOG “dff2_code.va”

.tran 1n 60u

.END
```

Active Device Netlist

```
*dff_sim
* Gateway 2.6.7.R Spice Netlist
Generator
* Simulation timestamp: 27-Sep-2007
10:59:38
* Schematic name: dff_sim
V2 clk GND PULSE(0 5 5u 0.1n 0.1n 5u
10u)
V3 VDD GND DC 5
V4 d GND PULSE(0 5 10u 0.1n 0.1n 10u
20u)
X1 clk d q qbar dff2
*
* Schematic name: dff2
.SUBCKT dff2 CLK D Q Q_BAR
X1 NET4 NET1 NET2 nand2
X2 NET2 CLK NET1 nand2
X4 NET3 D NET4 nand2
X5 NET1 Q_BAR Q nand2
X6 Q NET3 Q_BAR nand2
X7 NET1 CLK NET4 NET3 nand3
M1 GND Q GND GND nmos w=5u l=5u M=1
M2 GND Q_BAR GND GND nmos w=5u l=5u
M=1
.ENDS dff2
*
* Schematic name: nand2
.SUBCKT nand2 a b c
M1 NET3 a GND GND nmos w=0.5u l=0.35u
M=1
M2 NET1 b NET3 GND nmos w=0.5u
l=0.35u M=1
M3 NET1 a VDD VDD pmos w=2u l=0.35u
m=1
M4 NET1 b VDD VDD pmos w=2u l=0.35u
m=1
M5 NET2 NET1 VDD VDD pmos w=2u
l=0.35u m=1
M6 NET2 NET1 GND GND nmos w=0.5u
l=0.35u
+ M=1
M7 c NET2 GND GND nmos w=0.5u l=0.35u
M=1
M8 c NET2 VDD VDD pmos w=2u l=0.35u
m=1
.ENDS nand2
*
* Schematic name: nand3
.SUBCKT nand3 a b c d
```

```
M3 NET1 a VDD VDD pmos w=2u l=0.35u
m=1
M4 NET1 b VDD VDD pmos w=2u l=0.35u
m=1
M5 NET2 NET1 VDD VDD pmos w=2u
l=0.35u m=1
M7 d NET2 GND GND nmos w=0.5u l=0.35u
M=1
M8 d NET2 VDD VDD pmos w=2u l=0.35u
m=1
M10 NET1 c VDD VDD pmos w=2u l=0.35u
m=1
M11 NET2 NET1 GND GND nmos w=0.5u
l=0.35u
+ M=1
M12 NET1 b NET3 GND nmos w=0.5u
l=0.35u M=1
M13 NET3 a NET4 GND nmos w=0.5u
l=0.35u M=1
M14 NET4 c GND GND nmos w=0.5u
l=0.35u M=1
.ENDS nand3
*
* Global Nodes Declarations
.GLOBAL  clk GND VDD
* End of the netlist
*
* Markers to save
.SAVE ALL(V) V(q)
.model nmos nmos level=3
.model pmos pmos level=3
.tran 1n 60u
.END
```

Verilog-A Module

```
//D-Flip Flop
`include “discipline.h”

module dff2_code (d, clk, q, qbar);
input clk, d;
output q, qbar;
electrical d, clk, q, qbar;
// first and second stage states
electrical A1,A2;
parameter real high=5,low=0;
parameter real vtrans=(high+low)/2;
parameter real tdel=0n, trise=0.3n;
parameter real tfall=0.3n;
parameter real dt=trise/100;
// current state
integer LogicClk,LogicVin;
integer LogicA1,LogicA2;
// states for track-and-hold
integer Ka1,Ka2;

analog begin
// Convert signals to logic format:
LogicClk = (V(clk)>vtrans);
LogicVin = (V(d)>vtrans);
LogicA1 = (V(A1)>0.5);
LogicA2 = (V(A2)>0.5);

// measure input at crossing:
@(cross( V(clk)-vtrans,+1 ))
LogicClk=0;

// compute new logic states desired:
Ka1 = LogicClk? LogicA1:LogicVin;
Ka2 = LogicClk? LogicA1:LogicA2;

// remove AC component of signal
I(A1) <+ ddt(dt*V(A1));
I(A2) <+ ddt(dt*V(A2));
I(A1) <+ (V(A1)-Ka1);
I(A2) <+ (V(A2)-Ka2);

// output using transition statement
// inverse of output
V(q) <+ transition(Ka2?high:low,tdel,
trise,tfall);
V(qbar) <+ high+low - V(q);

end
endmodule
```