Gateway/SmartSpiceで Verilog-Aモデルを実行するための 新しいシンタックス

Verilog-Aモデルを用いるとシミュレーション時間を短縮できますが、最新のSmartSpiceの機能では、より簡単な操作でVerilog-Aモデルを回路に追加できるようになりました。新しいSmartSpiceでVerilog-Aモデルを追加するには、対象のインスタンスをサブサーキットとして呼び出し、Verilog-Aモジュールをインクルードするための.VERILOGステートメントを追加するだけです。

Verilog-Aは、トランジスタ・レベルの正確な回路を作成する用途にも使用できますが、回路の動作を確認する目的であれば、アクティブ・デバイスを含む回路の代わりに、ビヘイビア・モデルを使用できます。ビヘイビア・モデルを使用すると、従来は何日も要した大規模回路のシミュレーションを、わずか数分で実行できます。たとえば、図1に示すような単純なD型フリップフロップの回路例でも、単純なLevel=3トランジスタをVerilog-Aモデルに置き換えると、実行速度が10倍になります。より高位のモデルを使用した大規模で複雑な回路では、SPICEモデル使用の場合とVerilog-Aビヘイビア・モデル使用の場合の速度差は、優に何百倍にもなります。

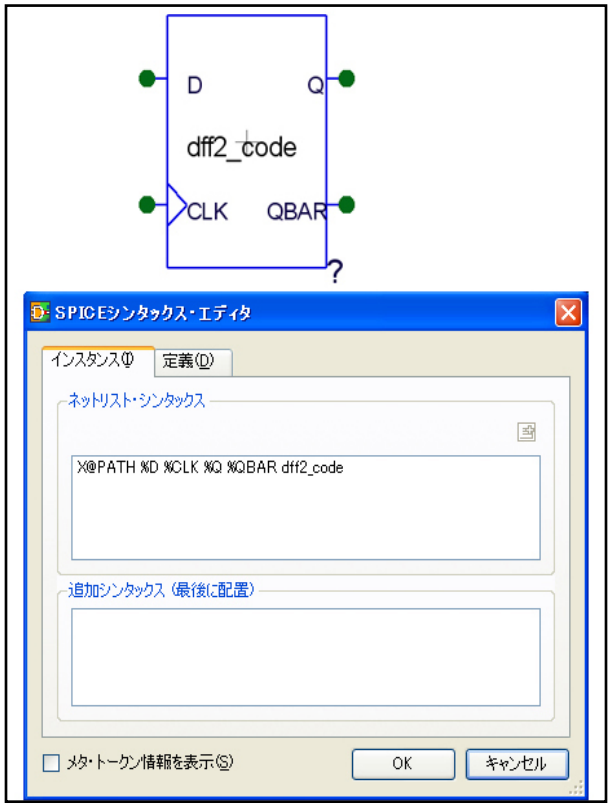

図1: SPICEシンタックス・エディタ

Verilog-Aデバイスを Gatewayに追加するには、シンボルを作成し、[SPICEシンタックス・エディタ]にサブサーキット呼び出しを追加します(図1参照)。シンタックスの先頭文字をXとすると、デバイスはサブサーキットとして指定されます。Verilog-Aモジュール内と同じ順番でピンを記述し、モジュール名を記述します。次に、コントロール・カードに、Verilog-Aモジュールを呼び出す.VERILOG呼び出しを追加します。Verilog-Aモジュールは、回路図と同じディレクトリにある必要はありませんが、異なるディレクトリにある場合は.VERILOG呼び出しにフルパスを記述する必要があります。Verilog-Aネットリスト例を参照してください。

図2では、2つのデバイス(Verilog-AとSPICE)の出力はほぼ同じに見えますが、図3のハイからローへの遷移を詳しく観察すると、Verilog-Aは単なるビヘイビア・モデルであり、トランジスタ・レベルで正確に一致するものではないことが分かります。しかしながら、コンセプトを検証する目的や、波形より遷移が大きな意味を持つようなブロックを置き換える目的では、この2つの結果は十分類似しているといえます。

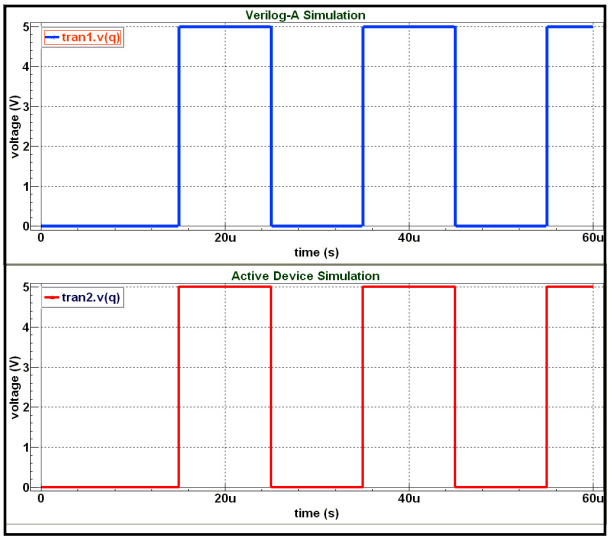

図2: 2種類のシミュレーション結果 #1 (Verilog-A vs SPICE)

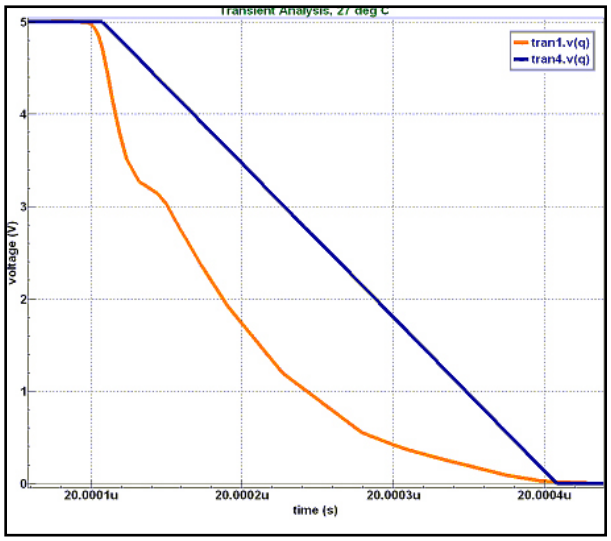

図3: 2つのシミュレーション結果 #2 (Verilog-A vs SPICE)

Verilogネットリスト

```
*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
```

アクティブ・デバイス・ネットリスト

```
*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モジュール

```
//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
```