BEHAVIORAL MODELING OF MUTI-TECHNOLOGICAL SYSTEMS WITH VHDL-AMS AND SIMULATING WITH SPICE

Some applications by using VamSpiceDesigner

Sabeur Jemmali, Ali Nehme and Jean-Jacques Charlot

GET/ENST, dept COMELEC
46, rue Barrault, 75013 Paris, France
sabeur.jemmali@enst.fr

Abstract

During the past years, a lot of works have been done about behavioral models and simulation tools. But a need of modeling strategy still remains. This paper presents a possible strategy of modeling multi-technological (electric and non-electric) systems by using VamSpiceDesigner, created by GET/ENST, containing free tools as a hierarchical schematics, a VHDL-AMS compiler and a SPICE simulator. In the introduction, the VamSpiceDesigner tool is described. The next sections shows three examples which represent different aspects of multi-technology: micro-electronics, mechatronics and colorimetry.

Introduction

A. Principle of the Schematic Design

The design of systems according to the top-down methods and bottom-up can be undertaken advantageously with VHDL-AMS while considering its capability to model at different abstraction level (figure 1) (1).

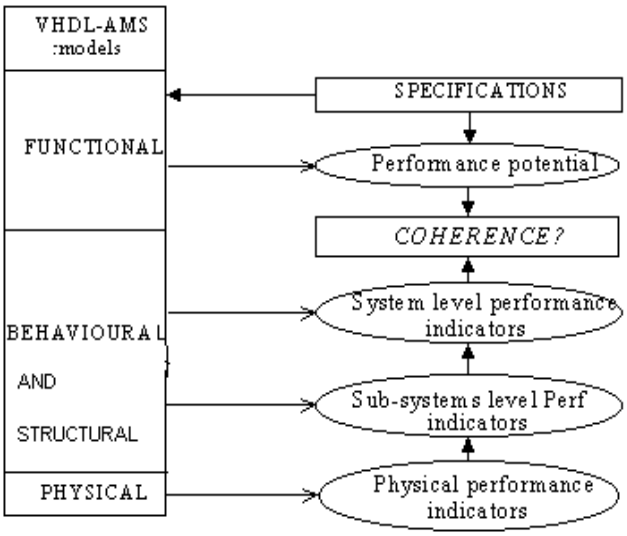

Fig. 1 : Top-down and bottom-up design methodology with VHDL-AMS

The suggested hierarchical schematic design methodology allows:

- to be able to go down, while following the process going down (*top-down*), in the various abstraction levels required by the project of design,
- to go up, while following the process going up (bottom-up).

It is summarized in figure 1:

An optimization between the rising and falling approaches to provide a coherent model could be found between its physical level and its functional level.

For that,

- The system is divided into various behavioral subsystems,
- Performance indicators for each level of modeling are provided,
- indicators impacts of performance are gradually reflected towards the complete system.

B. From VHDL-AMS to SPICE

The passage from VHDL-AMS to SPICE OPUS (2), the chosen SPICE simulator, is represented on the figure 2.

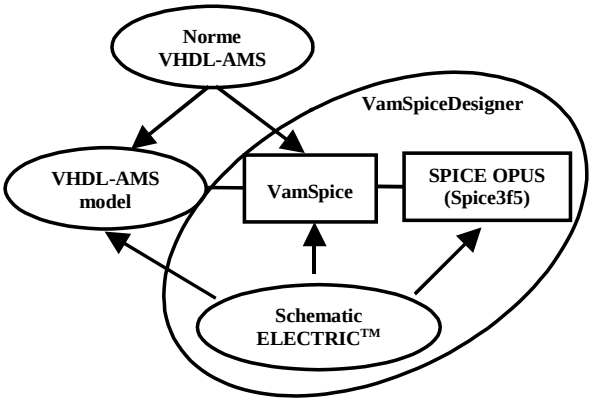

Fig. 2 : VamSpiceDesigner and the passage from VHDL-AMS to SPICE OPUS

.

C. VamSpiceDesigner

VamSpiceDesigner is a set of tools (figure 2) :

- Working on Windows (9x/ME/NT/2000/XP),
- Associating with VamSpice (3) compiler and with the SPICE OPUS simulator,
- A schematic ELECTRIC™ (4).

C.1. VamSpice compiler

VamSpice (**V**HDL-**AMS**/**SPICE**) (2) is a compiler that translates a VHDL-AMS model to SPICE OPUS, figure 2 shows this passage. The compiler/translator generates principally two files **cfunc.c** that contains the body of the VHDL-AMS model and **ifspec.c** that contains the ports and the generics declarations. Once these two files are compiled with the GNU tools and libraries and linked with SPICE

OPUS libraries, we obtain a DLL (Dynamic Link Library) file. This file can link up to SPICE OPUS.

C.2. SpiceOpus simulator (= SPICE3 + XSPICE + Nutmeg)
SpiceOpus (3) is a circuit simulator with optimization utilities. It is a recompilation of the original Berkeley's source code (SPICE3F5). XSPICE mixed-mode simulator was added to the Berkeley code. The Giorgia Tech Institute XSPICE code model feature was enhanced so that code models can be loaded from DLL files (.cm files). The DLL is generated from VamSpice compiler. The graphic part (Nutmeg) program was rewritten for adapted it to the Windows and Linux environment. The second part, the device model (.MODEL) is defined, through which, parameter values (VHDL-AMS generic) can be changed.
In a schematic view « {sch} », are drawn the outline of the circuit test of the component, while calling the icon that is defined in top. Next, power supplies, analyses types and outputs to be displayed are added and simulation can be run. ELECTIC^{TM} generates automatically an analogical netlist to be used by SPICE OPUS simulator.

C.3. Macro-component, icon and hierarchy
Macro-components can be obtained while transforming circuit in icon. An icon is a rectangle in which one cannot see a sub circuit, but only the name of the macro-component. The important property of this icon is to be able to appear as a component itself in circuits thanks to hierarchy order climbing or descending.

A micro-electronic application

This first example shows the use of VamSpiceDesigner for applying the methodology on a very simple case : MOSFET behavioral modeling. Figure 3 represents the upper view of the test circuit. The MOSFET equivalent circuit can be seen in typing successively on the MOSFET symbol and on Ctrl+D (for Down Hierarchy) (Figure 4: amos{sch}).

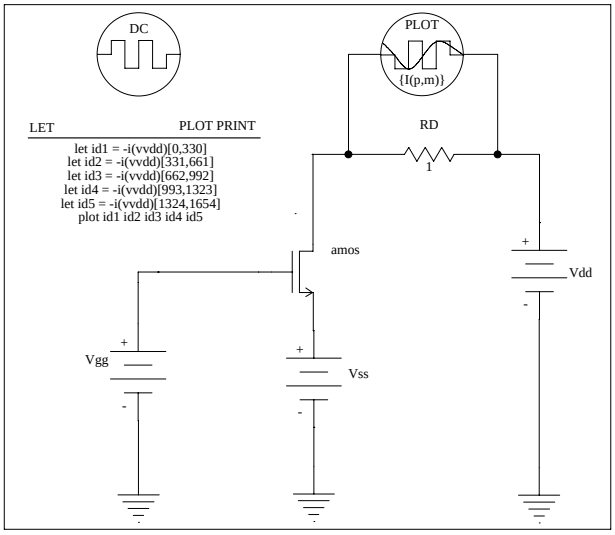

Fig. 3 : Test circuit of the functional MOSFET.

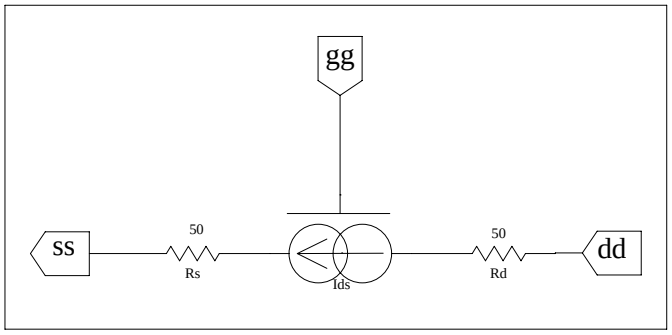

Fig. 4 : Equivalent circuit of the functional MOSFET.

Typing again on the current generator symbol and Ctrl+D makes appear the basic icon of the current generator (Figure 5 :ids{ic}). From this icon, the skeleton of a VHDL-AMS model file can be built automatically and has to be filled by the user as shown on figure 6. The equation of the drain current is here empirical giving to the MOSFET a functional behavior with few parameters for a future use in big circuits.

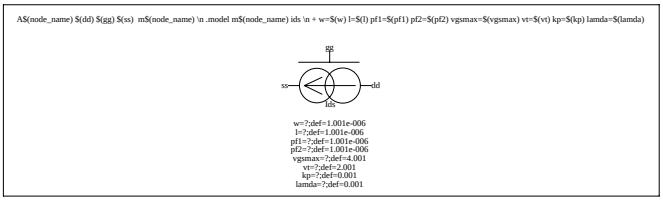

Fig. 5 : The basic icon of the current generator.

```
-- VHDL-AMS automatically generated from facet ids{ic}
ENTITY ids IS
 GENERIC (w : real:= 1.001e-006; kp : real:= 0.001; pf2 : real:=
 1.001e-006; l : real:= 1.001e-006; vt ::real:= 2.001; vgsmax : real:=
 4.001; lamda :real:= 0.001; pf1 :real:= 1.001e-006);
 PORT (TERMINAL dd, gg, ss: Electrical);
END ids;
ARCHITECTURE ids_BODY OF ids IS
-- ****************** tanh function ************************
  function ftanh(x1 : real) return real is
    variable y1:real;
  begin
    y1 := (exp(2.0 * x1) - 1.0) / (exp(2.0 * x1) + 1.0);
    return y1;
  end ftanh;
--************************************************************
quantity vds across ids through dd to ss;
quantity vgs across gg to ss;
BEGIN
  ids == pf1 * kp * (w/l) * vgs * ftanh(vgs) * (1.0 - exp(-pf2 * vds * (vgsmax +
vt  - vgs))) * (1.0 + lamda * vds);
END ids_BODY;
```

Fig. 6 : VHDL-AMS model of the MOSFET.

Going back to the test circuit by typing Ctrl+U (for Up Hierarchy) twice on the last icon ids{ic}, the SPICE OPUS simulation can then be run (figure 7 and figure 8).

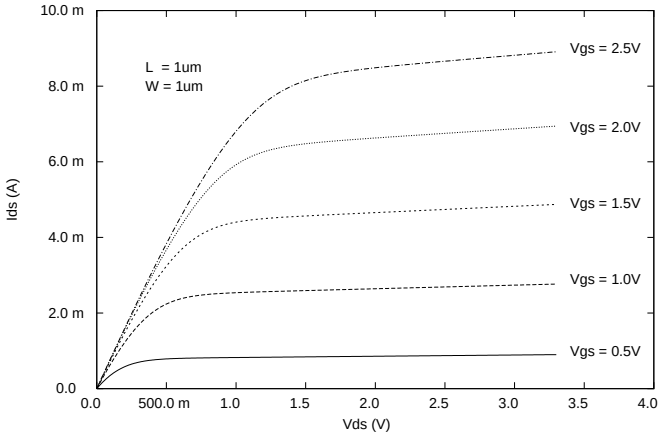

Fig. 7 : Static characteristic : Ids = F(Vds)

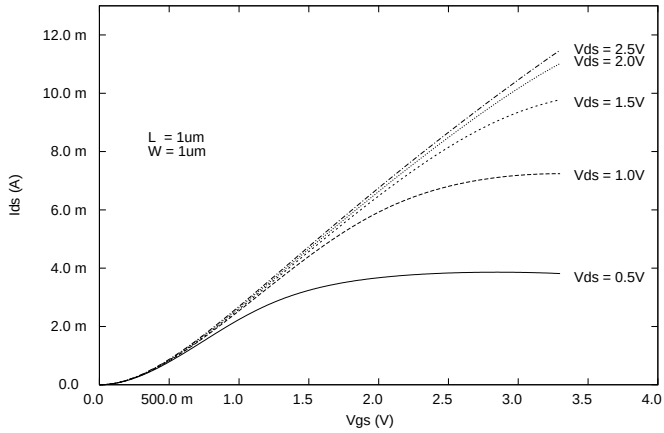

Fig. 8 : Static characteristic : Ids = F(Vgs)

A mechatronic application

This example is inspired from the one extracted from (5). It analyses the behavior of the elevation of a seat in a car. Figure 10 represents the bloc diagram containing the different blocks of the system: DC motor, gear, velocity converter and pinion-rack. Figures 11, 12 show the icon corresponding to the DC motor and the pinion-rack. From pinion-rack icon the skeleton of the VHDL-AMS model is automatically generated and completed (Figure 9).

```
-- VHDL-AMS automatically generated from facet pinionrack{ic}
ENTITY pinionrack IS
 GENERIC (tau :real:= 0.1; posmin :real:= 0.02; posmax :real:= 0.1;
 radius :real:= 0.1);
 PORT (TERMINAL pinion, gndmec: Mecanical; QUANTITY position: OUT
 Real);
END pinionrack;
ARCHITECTURE pinionrack_BODY OF pinionrack IS
 quantity apos, pos1 : real;
 quantity piangle across pos through pinion to gndmec;
BEGIN
 apos == radius * piangle;
 pos1 == 0.9*posmax;
 if (apos < posmin) USE
    pos == posmin;
 else if (apos > pos1) use
        pos == 0.9 * posmax + 0.1 * posmax * (1.0 - exp((0.9 * posmax - apos) /
tau));
      else  pos == apos;
      end use;
 end use;
 position == pos;
END pinionrack_BODY;
```

Fig. 9 : VHDL-AMS model of pinion-rack

The position of the seat is calculated and limited by a minimum position and a maximum position (which is reached smoothly with an exponential expression). Figure 13 presents the elevation of the seat versus the voltage applied to the DC motor.

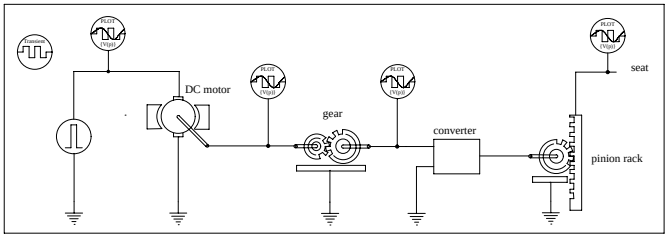

Fig. 10 : Test circuit of a seat elevator

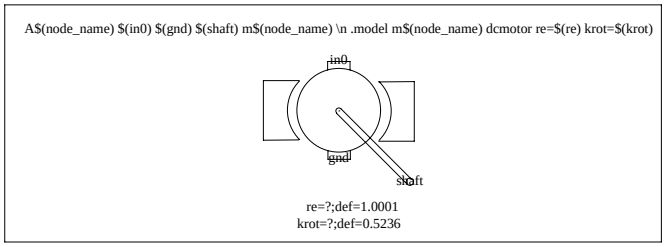

Fig. 11 : DC motor Icon

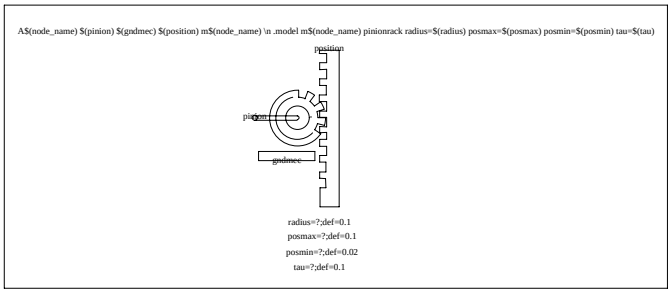

Fig. 12 : Pinion-rack icon

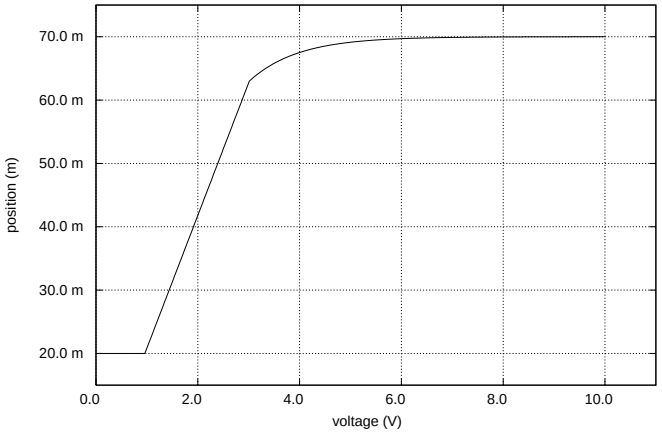

Fig. 13 : Seat elevation position (voltage)

A colorimetric application

As an heteroclite and unexpected example, this one has been inspired by our previous work presented in (6). A colored object (C1 spectre) is illuminated by a specified illuminant (S) and observed through fog of k density at a defined distance. The colorimetric system is set up for determining the color rendered in a hazy environment (Figure 14). It contains the following modules:

1) Illuminant module contains information on the type of illuminant, here the normalized D65 illuminant
2) x*fogxyz* module calculates the effect of fog and distance on the object reflectance for each basic stimulus xs, ys and zs
3) *trisxyz* module uses the analogy of color with electricity as discussed in (6) for calculating the tristimili X, Y and Z, here by using inductors. It means that an integration is done for getting them.
4) *cciexyz* module transforms tristimuli X, Y and Z in chromatic coordinates x, y and z
5) *XYZTORGB* module transforms tristimuli in R, G and B fundamental colors according to the screen specification on which rendered color has to be displayed
6) *WRITE* module makes writing results in a file named rawspice.raw that is used for plotting results by SPICE NUTMEG and as inputs of a program for displaying effectively rendered colors on a screen (7).

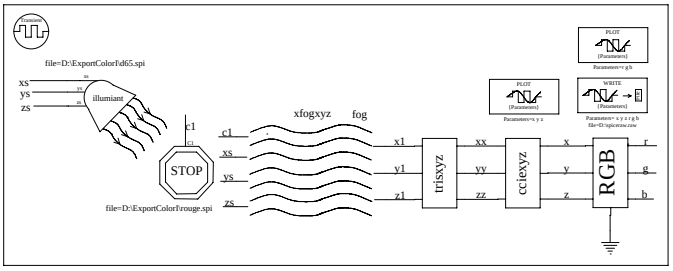

Fig. 14 : Bloc diagram of the colorimetric system

The same procedure than the previous one is used for getting icons (Figure 15) from which VHDL-AMS skeleton file are automatically generated to be filled by the user. For instance, xfog models of xfogxyz module is given by Figure 16.

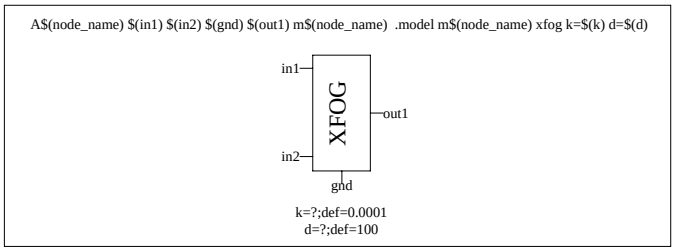

Fig. 15 : XFOG icon.

```
-- VHDL-AMS automatically generated from facet xfog{ic}
ENTITY xfog IS
 GENERIC (d :real:= 100.0; k :real:= 0.0001);
 PORT (TERMINAL in1, in2, gnd: Electrical; QUANTITY out1: OUT Real);
END xfog;
ARCHITECTURE xfog_BODY OF xfog IS
 quantity vinxyz across in1 to gnd;
 quantity vinr across in2 to gnd;
BEGIN
  out1 == vinxyz * (1.0 - (1.0 - vinr) * exp(-k*d));
END xfog_BODY;
```

Fig. 16 : Fog model

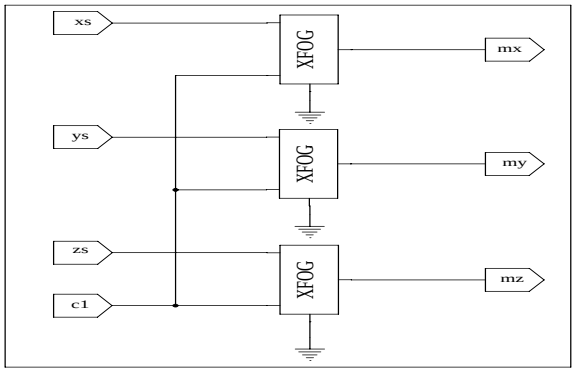

Fig. 17 : Fog model for tristimuli X, Y and Z

To rendered colors on a screen depends on the color monitor used (8). The characteristics of a color monitor are determined by the colors of its guns, its red, green, and blue primaries denoted R, G, and B and determined by primary coordinates chromaticities (x_R, y_R), (x_G, y_G), and (x_B, y_B). In this example, we choice NTSC (9) (National Television System Committee) as color system and 1.0 for gamma correction. Gamma correction represents a numerical parameter that describes the nonlinearity of intensity reproduction. Figure 18 shows the position of the final point of the curve representing the rendered color inside of the "CIE horseshoe (10)". For that case, the fog factor was fixed at $k=0.0004m^{-1}$ and the distance was varied from 1 to 1000m (in figure 18, the distance is fixed to 1000m) the rendered color of the object, originally red, is quite pink . This corresponds to the well known phenomena: color whitening through fog observation .

Conclusions

This paper describe a strategy of modeling multi-technological systems by using VamSpiceDesigner© GET in three examples. In the first example, a micro-electronic application is described by modeling a behavioral and functional MOSFET in order to present the strategy. In the second, a mechatronic application is presented : the elevation of a seat in a car is commanded by a voltage. The third example shows the effect of fog and distance is modeled on the rendered color of an colored object. These three example allow to show the big variety of multi-technological domains that can be modeled by VHDL-AMS.

References

(1) S. Jemmali and J.-J. Charlot, "VamSpice Designer, a hierarchical schematic design tool of multi-technological systems based on VHDL-AMS and SPICE", MIXDES'03 June 2003, Lodz, Poland

(2) J.-J. Charlot and S. Jemmali, VamSpice © GET/ENST Paris 2002

(3) SPICE OPUS, The University of Ljubljana, Slovenia, http://www.fe.unilj.si/spice/welcome.html

(4) ELECTRIC™, http://www.staticfreesoft.com

(5) H. A. Mantooth, M. Fiegenbaum, "Modeling with an Analog Hardware Description Language", Kluwer Academic Publishers, 1995.

(6) J.-J. Charlot, E. Barker, O. Alali, J.-F. Charlot, "Color Rendering in a Hazy Environment" : Simulation with SPICE3F5/VHDL-AMS", BMAS'98, October 1998, Orlando, USA

(7) S.Jemmali and J.-J Charlot, "Colorimetric v1.03",Created at ENST 2003

(8) Where's purple ? Or, how to plot colours properly on a computer screen, http://casa.colorado.edu/ãjsh/colour/rainbow.html

(9) efg's Computer Lab, Chine Transation by Hector Xiang, "Chromaticity Diagrams Lab Report", http://www.efg2.com/

(10) Commission Internationale de l'Éclairage, http://www.cie.co.at/cie/

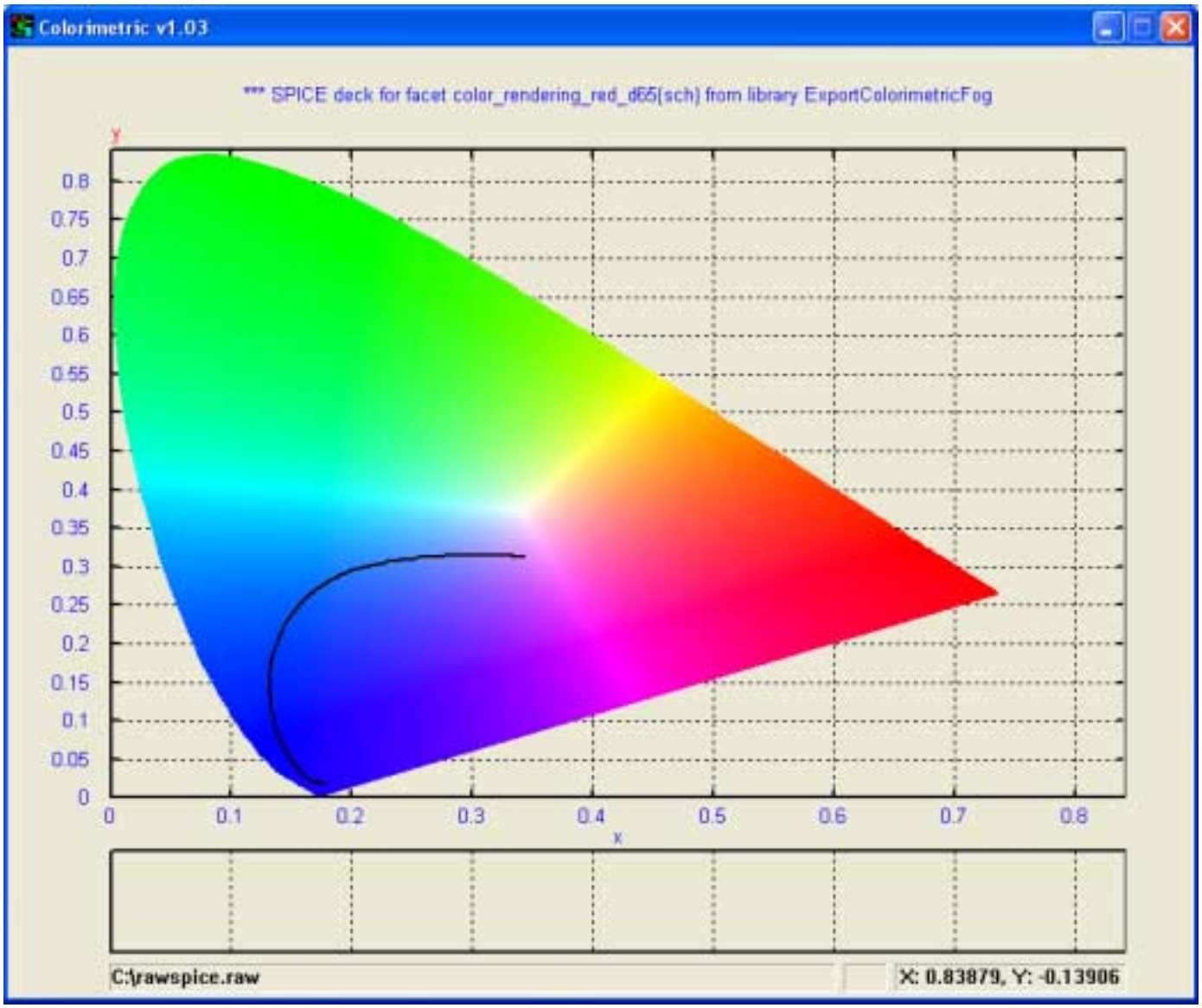

Fig. 18 : Simulation results of rendered colors.
The “horseshoe” is CIE (10) 1931 chromaticity diagram – 1931 2° observer.