Verilog-A for Memristor Models

Shahar Kvatinsky*, Keren Talisveyberg*, Dmitry Fliter*, Eby G. Friedman**, Avinoam Kolodny*, and Uri C. Weiser*

**Department of Electrical Engineering*
Technion – Israel Institute of Technology
Haifa 32000, ISRAEL

***Department of Electrical and Computer Engineering*
University of Rochester
Rochester, New York 14627, USA

Abstract — **Memristors are novel devices, which can be used in applications such as memory, logic, and neuromorphic systems. Several models for memristors have been developed – the linear ion drift model, the nonlinear ion drift model, the Simmons tunnel barrier model, and the ThrEshold Adaptive Memristor (TEAM) model. In this technical report a Verilog-A implementation for these models and the relevant window functions is presented, suitable for EDA tools, such as SPICE.**

Keywords – memristor; memristive systems, SPICE, Verilog-A;

I. INTRODUCTION

Memristors are passive two-port elements with variable resistance (also known as a memristance) [1]. Changes in the memristance depend upon the history of the device (*e.g.*, the memristance may depend on the total charge passed through the device, or alternatively, on the integral over time of the applied voltage between the ports of the device).

To use EDA tools for simulations of memristor-based circuits, a specific memristor model is needed. Several memristor models have been proposed. In this technical report a Verilog-A code for different memristor models is presented. A complementary GUI MATLAB program is also available in [3], useful for initial work with these memristor models.

II. MEMRISTOR MODELS

All the memristor models have been implemented in the Verilog-A model are presented in [2]. In this technical report only a brief description is provided. The equations and main characteristics of the memristor models are listed in Table 1 and 2.

A. Linear Ion Drift Model

In the linear ion drift model, two resistors are connected in series, one resistor represents the high concentration of dopants region (high conductance) and the second resistor represents the oxide region (low conductance). It is also assumed a linear ion drift in a uniform field and that the ions have equal average ion mobility μ_V.

B. Nonlinear Ion Drift Model

The nonlinear ion drift model is assumed a voltage-controlled memristor with nonlinear dependence between the voltage and the internal state derivative. In this model, the state variable w is a normalized parameter within the interval [0, 1]. This model also assumes asymmetric switching behavior.

C. Simmons Tunnel Barrier Model

This model assumes nonlinear and asymmetric switching behavior due to an exponential dependence of the movement of the ionized dopants, namely, changes in the state variable. In this model, rather than two resistors in series as in the linear drift model, there is a resistor in series with an electron tunnel barrier. In this model, the state variable x is the Simmons tunnel barrier width.

D. ThrEshold Adaptive Memristor (TEAM) Model

The TEAM model is a general memristor model; assume that the memristor has a current threshold and polynomial dependence between the memristor current and the internal state drift derivative. The current-voltage relationship can be in a linear or exponential manner. It is possible to fit the TEAM model to the Simmons tunnel barrier model or to any different memristor model and gain a more efficient computational time.

III. WINDOW FUNCTIONS

To force the bounds of the device and to add nonlinear behavior close to these bounds, several window functions have implemented in the Verilog-A model. The implemented window functions are: Jogelkar, Biolek, Prodromakis, and TEAM (named Kvatinsky in the Verilog-A model). The window functions are presented in [2] and their main characteristics are listed in Table 3.

TABLE 1. THE CHARACTERISTICS OF THE MEMRISTOR MODELS (FURTHER DESCRIPTION IN [2])

Model	Linear ion drift	Nonlinear ion drift	Simmons tunneling barrier	TEAM
State variable	$0 \le w \le D$ Doped region physical width	$0 \le w \le 1$ Doped region normalized width	$a_{off} \le x \le a_{on}$ Undoped region width	$x_{on} \le x \le x_{off}$ Undoped region width
Control mechanism	Current controlled	Voltage controlled	Current controlled	Current controlled
Current-voltage relationship and memristance deduction	Explicit	I-V relationship – explicit Memristance deduction - ambiguous	Ambiguous	Explicit
Matching memristive system definition	Yes	No	No	Yes
Generic	No	No	No	Yes
Accuracy comparing practical memristors	Lowest accuracy	Low accuracy	Highest accuracy	Sufficient accuracy
Threshold exists	No	No	Practically exists	Yes

TABLE 2. THE MATHEMATICAL DESCRIPTION OF THE VERILOG-A MEMRISTOR MODEL (FURTHER DESCRIPTION IN [3])

Model	Current-voltage relationship	State variable derivative
Linear ion drift	$v(t)=\left(R_{ON}\frac{w(t)}{D}+R_{OFF}\left(1-\frac{w(t)}{D}\right)\right)\cdot i(t)$	$\frac{dw}{dt}=\mu_v\frac{R_{ON}}{D}i(t)$
Nonlinear ion drift	$i(t)=w(t)^n\beta\sinh\left(\alpha v(t)\right)+\chi\left[\exp\left(\gamma v(t)\right)-1\right]$	$\frac{dw}{dt}=a\cdot f(w)\cdot v(t)^m$
Simmons tunneling barrier	$v(t)=\left[R_{ON}+\frac{R_{OFF}-R_{ON}}{x_{off}-x_{on}}\left(x-x_{on}\right)\right]\cdot i(t)$ or $v(t)=R_{ON}e^{\frac{\lambda}{x_{off}-x_{on}}(x-x_{on})}\cdot i(t)$ Note that this is different than original Simmons tunneling barrier	$\frac{dx(t)}{dt}=\begin{cases} c_{off}\sinh\left(\frac{i}{i_{off}}\right)\exp\left[-\exp\left(\frac{x-a_{off}}{w_c}-\frac{\lvert i\rvert}{b}\right)-\frac{x}{w_c}\right], & i>0 \\ c_{on}\sinh\left(\frac{i}{i_{on}}\right)\exp\left[-\exp\left(-\frac{x-a_{on}}{w_c}-\frac{\lvert i\rvert}{b}\right)-\frac{x}{w_c}\right], & i<0 \end{cases}$
TEAM	$v(t)=\left[R_{ON}+\frac{R_{OFF}-R_{ON}}{x_{off}-x_{on}}\left(x-x_{on}\right)\right]\cdot i(t)$ or $v(t)=R_{ON}e^{\frac{\lambda}{x_{off}-x_{on}}(x-x_{on})}\cdot i(t)$	$\frac{dx(t)}{dt}=\begin{cases} k_{off}\cdot\left(\frac{i(t)}{i_{off}}-1\right)^{\alpha_{off}}\cdot f_{off}(x), & 0<i_{off}<i \\ k_{on}\cdot\left(\frac{i(t)}{i_{on}}-1\right)^{\alpha_{on}}\cdot f_{on}(x), & i<i_{on}<0 \\ 0, & otherwise \end{cases}$

TABLE 3. COMPARISON OF DIFFERENT WINDOW FUNCTIONS (FURTHER DESCRIPTION IN [2])

Function	Jogelkar	Biolek	Prodromakis	TEAM
$f(x)/f(w)$	$f(w) = 1-(2w/D-1)^{2p}$	$f(w) = 1-(w/D-stp(-i))^{2p}$	$f(w)=j(1-[(w-0.5)^2+0.75]^p)$	$f_{on,off}=\exp[-\exp(\lvert x-x_{on,off}\rvert/w_c)]$
Symmetric	Yes	Yes	Yes	Not necessarily
Resolve boundary conditions	No	Discontinuities	Practically yes	Practically yes
Impose nonlinear drift	Partially	Partially	Partially	Yes
Scalable $f_{max} < 1$	No	No	Yes	No
Fits memristor model	Linear/nonlinear ion drift/TEAM	Linear/nonlinear ion drift/TEAM	Linear/nonlinear ion drift/TEAM	TEAM for Simmons tunneling barrier fitting

IV. VERILOG-A CODE

```
///////////////////////////////////////////////////
// VerilogA model for memristor
//
// kerentalis@gmail.com
// Dimafliter@gmail.com
// skva@tx.technion.ac.il
//
// Technion - Israel institute of technology
// EE Dept. December 2011
//
///////////////////////////////////////////////////

`include "disciplines.vams"
`include "constants.h"

// define meter units for w parameter
nature distance
  access = Metr;
  units = "m";
  abstol = 0.01n;
endnature

discipline Distance
  potential distance;
enddiscipline


module Memristor(p, n,w_position);
  input p;//positive pin
  output n;//negative pin
  output w_position;// w-width pin

  electrical p, n,gnd;
  Distance w_position;
  ground gnd;

   parameter real model = 0;
// define the model:
// 0 - Linear Ion Drift;
// 1 - Simmons Tunnel Barrier;
// 2 - Team model;
// 3 - Nonlinear Ion Drift model

   parameter real window_type=0;
// define the window type:
// 0 - No window;
// 1 - Jogelkar window;
// 2 - Biolek window;
// 3 - Prodromakis window;
// 4 - Kvatinsky window (Team model only)

   parameter real dt=0;
// user must specify dt same as max step size in
// transient analysis & must be at least 3 orders
//smaller than T period of the source

   parameter real init_state=0.5;
// the initial state condition [0:1]


///////////// Linear Ion Drift model //////////////

 //parameters definitions and default values
    parameter real Roff = 200000;
    parameter real Ron = 100;
    parameter real D = 3n;
    parameter real uv = 1e-15;
    parameter real w_multiplied = 1e8;
// transformation factor for w/X width
// in meter units
    parameter real p_coeff = 2;
     // Windowing function coefficient

     parameter real J = 1;
     // for prodromakis Window function

     parameter real p_window_noise=1e-18;
     // provoke the w width not to get stuck at
     // 0 or D with p window

     parameter real threshhold_voltage=0;

     // local variables
     real w;
     real dwdt;
     real w_last;
     real R;
     real sign_multply;
     real stp_multply;
     real first_iteration;


////////// Simmons Tunnel Barrier model ///////////

   //parameters definitions and default values
  //for Simmons Tunnel Barrier model
    parameter real c_off = 3.5e-6;
    parameter real c_on = 40e-6;
    parameter real i_off = 115e-6;
    parameter real i_on = 8.9e-6;
    parameter real x_c = 107e-12;
    parameter real b = 500e-6;
    parameter real a_on = 2e-9;
    parameter real a_off = 1.2e-9;

    // local variables
    real x;
    real dxdt;
    real x_last;


/////////////////TEAM model////////////////////

    parameter real K_on=-8e-13;
    parameter real K_off=8e-13;
    parameter real Alpha_on=3;
    parameter real Alpha_off=3;
    parameter real IV_relation=0;
     // IV_relation=0 means linear V=IR.
     // IV_relation=1 means nonlinear V=I*exp{..}
    parameter real x_on=0;
    parameter real x_off=3e-09; // equals D

     // local variables
    real lambda;


////////////Nonlinear Ion Drift model ////////////

    parameter real alpha = 2;
    parameter real beta  = 9;
    parameter real c     = 0.01;
    parameter real g     = 4;
    parameter real N     = 14;
    parameter real q     = 13;
    parameter real a     = 4;

 analog function integer sign;
  //Sign function for Constant edge cases
     real arg; input arg;
     sign = (arg >= 0 ? 1 : -1 );
 endfunction

 analog function integer stp;         //Stp function
     real arg; input arg;
     stp = (arg >= 0 ? 1 : 0 );
 endfunction


/////////////////// MAIN //////////////////////

 analog begin

    if(first_iteration==0) begin
        w_last=init_state*D;
     // if this is the first iteration,
     //start with w_init
```

```
        x_last=init_state*D;
     // if this is the first iteration,
     // start with x_init
     end

/////////////Linear Ion Drift model //////////////

if (model==0) begin // Linear Ion Drift model

     dwdt =(uv*Ron/D)*I(p,n);

      //change the w width only if the
      // threshhold_voltage permits!
       if(abs(I(p,n))<threshhold_voltage/R) begin
            w=w_last;
        dwdt=0;
       end

     // No window
     if ((window_type==0)|| (window_type==4)) begin

        w=dwdt*dt+w_last;

     end // No window

   // Jogelkar window
      if (window_type==1) begin

        if (sign(I(p,n))==1) begin
            sign_multply=0;
            if(w==0) begin
            sign_multply=1;
            end
        end
        if (sign(I(p,n))==-1) begin
                sign_multply=0;
                if(w==D) begin
                    sign_multply=-1;
                    end
        end

        w=dwdt*dt*(1-pow(2*w/D-
1,2*p_coeff))+w_last+sign_multply*p_window_noise;

     end // Jogelkar window


     // Biolek window
     if (window_type==2) begin

        if (stp(-I(p,n))==1) begin
            stp_multply=1;
        end
        if (stp(-I(p,n))==0) begin
                stp_multply=0;
                end

      w=dwdt*dt*(1-pow(w/D-
stp_multply,2*p_coeff))+w_last;

     end // Biolek window


     // Prodromakis window
     if (window_type==3) begin

        if (sign(I(p,n))==1) begin
            sign_multply=0;
            if(w==0) begin
            sign_multply=1;
            end
        end
        if (sign(I(p,n))==-1) begin
                sign_multply=0;
                if(w==D) begin
                    sign_multply=-1;
                    end
        end

        w=dwdt*dt*J*(1-pow(pow(w/D-
0.5,2)+0.75,p_coeff))+w_last+sign_multply*p_window_n
oise;

     end  // Prodromakis window

     if (w>=D) begin
         w=D;
                 dwdt=0;
     end

     if (w<=0) begin
         w=0;
                 dwdt=0;
     end


        //update the output ports(pins)
        R=Ron*w/D+Roff*(1-w/D);
        w_last=w;
        Metr(w_position) <+ w*w_multiplied;
        V(p,n) <+ (Ron*w/D+Roff*(1-w/D))*I(p,n);
        first_iteration=1;

end // end Linear Ion Drift model


///////// Simmons Tunnel Barrier model ///////////


if (model==1) begin // Simmons Tunnel Barrier model

     if (sign(I(p,n))==1) begin

        dxdt =c_off*sinh(I(p,n)/i_off)*exp(-
exp((x_last-a_off)/x_c-abs(I(p,n)/b))-x_last/x_c);
     end


     if (sign(I(p,n))==-1) begin
        dxdt =c_on*sinh(I(p,n)/i_on)*exp(-exp((a_on-
x_last)/x_c-abs(I(p,n)/b))-x_last/x_c);

     end


         x=x_last+dt*dxdt;


      if (x>=D) begin
            x=D;
            dxdt=0;
      end
      if (x<=0) begin
            x=0;
            dxdt=0;
      end


        //update the output ports(pins)
        R=Ron*(1-x/D)+Roff*x/D;
        x_last=x;
        Metr(w_position) <+ x/D;
        V(p,n) <+ (Ron*(1-x/D)+Roff*x/D)*I(p,n);
        first_iteration=1;

end // end Simmons Tunnel Barrier model

//////////////////// TEAM model /////////////////


if (model==2) begin // TEAM model

     if (I(p,n) >= i_off) begin
         dxdt =K_off*pow((I(p,n)/i_off-1),Alpha_off);
     end

     if (I(p,n) <= i_on) begin
         dxdt =K_on*pow((I(p,n)/i_on-1),Alpha_on);
     end
```

```
        if ((i_on<I(p,n)) && (I(p,n)<i_off)) begin
    dxdt=0;
    end


    // No window
    if (window_type==0) begin

    x=x_last+dt*dxdt;

    end // No window

    // Jogelkar window
    if (window_type==1) begin

        x=x_last+dt*dxdt*(1-pow((2*x_last/D-
1),(2*p_coeff)));

    end // Jogelkar window


    // Biolek window
    if (window_type==2) begin

        if (stp(-I(p,n))==1) begin
            stp_multply=1;
        end
        if (stp(-I(p,n))==0) begin
            stp_multply=0;
            end

        x=x_last+dt*dxdt*(1-pow((x_last/D-
stp_multply),(2*p_coeff)));

    end  // Biolek window


    // Prodromakis window
    if (window_type==3) begin

            x=x_last+dt*dxdt*J*(1-
pow((pow((x_last/D-0.5),2)+0.75),p_coeff));

    end  // Prodromakis window


    //Kvatinsky window
    if (window_type==4) begin

           if (I(p,n) >= 0) begin
           x=x_last+dt*dxdt*exp(-exp((x_last-
a_off)/x_c));
           end

       if (I(p,n) < 0) begin
           x=x_last+dt*dxdt*exp(-exp((a_on-
x_last)/x_c));
       end

        end // Kvatinsky window

    if (x>=D) begin
    dxdt=0;
    x=D;
    end

    if (x<=0) begin
    dxdt=0;
    x=0;
    end

       lambda = ln(Roff/Ron);

     //update the output ports(pins)
        x_last=x;
        Metr(w_position) <+ x/D;

     if (IV_relation==1) begin

         V(p,n) <+ Ron*I(p,n)*exp(lambda*(x-
x_on)/(x_off-x_on));

      end

      else if (IV_relation==0) begin

         V(p,n) <+ (Roff*x/D+Ron*(1-x/D))*I(p,n);

      end

     first_iteration=1;

end // end Team model


//////////// Nonlinear Ion Drift model ////////////


if (model==3) begin // Nonlinear Ion Drift model

    if (first_iteration==0) begin
        w_last=init_state;
    end

    dwdt = a*pow(V(p,n),q);


  // No window
    if ((window_type==0) || (window_type==4)) begin
        w=w_last+dt*dwdt;
        end // No window


  // Jogelkar window
    if (window_type==1) begin

        w=w_last+dt*dwdt*(1-pow((2*w_last-
1),(2*p_coeff)));

    end // Jogelkar window


  // Biolek window
    if (window_type==2) begin

        if (stp(-V(p,n))==1) begin
            stp_multply=1;
        end
        if (stp(-V(p,n))==0) begin
                stp_multply=0;
            end

        w=w_last+dt*dwdt*(1-pow((w_last-
stp_multply),(2*p_coeff)));

    end // Biolek window


  // Prodromakis window
    if (window_type==3) begin

            w=w_last+dt*dwdt*J*(1-pow((pow((w_last-
0.5),2)+0.75),p_coeff));

    end  // Prodromakis window

    if (w>=1) begin
        w=1;
        dwdt=0;
    end

    if (w<=0) begin
        w=0;
        dwdt=0;
    end

    //change the w width only if the
    // threshhold_voltage permits!
```

```
        if(abs(V(p,n))<threshhold_voltage) begin
            w=w_last;
        end

      //update the output ports(pins)
       w_last=w;
           Metr(w_position) <+ w;
           I(p,n) <+
pow(w,N)*beta*sinh(alpha*V(p,n))+c*(exp(g*V(p,n))-
1);
           first_iteration=1;

end // end Nonlinear Ion Drift model


  end   // end analog

endmodule
```

ACKNOWLEDGMENTS

This work was partially supported by Hasso Plattner Institute, by the Advanced Circuit Research Center at the Technion, and by Intel grant no. 864-737-13.

REFERENCES

[1] L. O. Chua, “Memristor – the Missing Circuit Element,” *IEEE Transactions on Circuit Theory*, Vol. 18, No. 5, pp. 507-519, September 1971.

[2] S. Kvatinsky, E. G. Friedman, A. Kolodny, and U. C. Weiser, "TEAM: ThrEshold Adaptive Memristor Model," submitted to *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2012.

[3] http://webee.technion.ac.il/people/skva/memristor.htm