Models of Memristors for SPICE Simulations

Shahar Kvatinsky, Keren Talisveyberg, Dmitry Fliter, Avinoam Kolodny, and Uri C. Weiser
Department of Electrical Engineering
Technion – Israel Institute of Technology
Haifa 32000, ISRAEL
skva@tx.technion.ac.il

Eby G. Friedman
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, analog circuits, and neuromorphic systems. Several memristor technologies have been developed such as ReRAM (Resistive RAM), MRAM (Magnetoresistance RAM), and PCM (Phase Change Memory). To design circuits with memristors, the behavior of the memristor needs to be described by a mathematical model. While the model for memristors should be sufficiently accurate as compared to the behavior of physical devices, the model must also be computationally efficient. Several models for memristors have been proposed – the linear ion drift model, the nonlinear ion drift model, the Simmons tunnel barrier model, and the ThrEshold Adaptive Memristor (TEAM) model. In this paper, the different memristor models are described and a Verilog-A implementation for these models, including the relevant window functions, are presented. These models are suitable for EDA tools such as SPICE.**

***Index Terms*— memristor, memristive systems, SPICE, Verilog-A, TEAM.**

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). Memristive systems [2] are an extension to memristors, where a current-controlled time-invariant memristive device is represented by

$$\frac{dw}{dt} = f(w,i), \tag{1}$$

$$v(t) = R(w,i)\cdot i(t), \tag{2}$$

where w is an internal state variable, i(t) is the current of the memristive device, $v(t)$ is the voltage across the memristive device, $R(w, i)$ is the memristance, and t is time. The terms memristor and memristive systems are often used interchangeably to describe memristive devices.

Memristors can be used in applications such as memory, logic, analog circuits, and neuromorphic systems. A memristor offers several advantages as compared to standard memory technologies: nonvolatility, good scalability, effectively no leakage current, and compatibility with CMOS technology, both electrically and in terms of manufacturing. Several memristor technologies have been developed such as ReRAM (Resistive RAM), MRAM (Magnetoresistance RAM), and PCM (Phase Change Memory).

To design circuits with memristors, the behavior of the memristor needs to be described by a mathematical model. While the model for memristors should be sufficiently accurate as compared to the behavior of physical devices, it must also be computationally efficient. It is also desirable for the model to be simple, intuitive, and closed-form, as well as general so that it can be tuned to suit different technologies of memristors. Several memristor models have been proposed: the linear ion drift model, the nonlinear ion drift model, the Simmons tunnel barrier model, and the ThrEshold Adaptive Memristor (TEAM) model. In this paper, the different memristor models are described and a Verilog-A code for these models and the relevant window functions are presented. These models are suitable for EDA tools such as SPICE.

II. Memristor Models

All of the memristor models which have been implemented in the Verilog-A model are described in [3]. In this paper, only a brief description of these models is provided. The basic equations and the main characteristics of the memristor models are listed in Table 1. A user manual to this Verilog-A model is provided in [4].

A. Linear Ion Drift Model

In the linear ion drift model [5], 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). A linear ion drift in a uniform field is also assumed, where the ions have equal average ion mobility μ_V. This model exhibits the definition of the original memristor in [1], but is inaccurate as compared to physical memristive devices.

B. Nonlinear Ion Drift Model

In the nonlinear ion drift model [6], a voltage-controlled memristor exhibiting a nonlinear dependence between the voltage and the internal state derivative is assumed. In this model, the state variable w is a normalized parameter within the interval [0, 1]. This model also assumes an asymmetric switching behavior.

C. Simmons Tunnel Barrier Model

The Simmons tunnel barrier model [7] assumes nonlinear and asymmetric switching behavior due to an exponential dependence of the movement of the ionized dopants, namely, changes in the internal 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. The state variable *x* is the Simmons tunnel barrier width.

D. ThrEshold Adaptive Memristor (TEAM) Model

The TEAM model [3] is a general memristor model. In this model, a current threshold and tunable nonlinear (polynomial) dependence between the current and the derivative of the internal state variable are assumed. The current-voltage relationship can be in a linear or an 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 with sufficient accuracy.

III. Window Functions

To maintain the physical bounds of the device and add nonlinear behavior close to these physical bounds, several window functions are implemented in the Verilog-A model. These window functions are: Jogelkar [8], Biolek [9], Prodromakis [10], and TEAM (named Kvatinsky in the Verilog-A model). The window functions and main properties are listed in Table 2.

IV. Verilog-A Code

```
`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 for linear model
    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 treshhold_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
    parameter real f_off = 3.5e-6;
    parameter real f_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))<treshhold_voltage/R) begin
          w=w_last;
                dwdt=0;
     end
     if ((window_type==0)|| (window_type==4)) begin // No
     // window
      w=dwdt*dt+w_last;
          end // No window
          if (window_type==1) begin // Jogelkar window
                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
          if (window_type==2) begin // Biolek window
                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
          if (window_type==3) begin // Prodromakis window
                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_noise;
          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      =f_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    =    f_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
          if (window_type==0) begin // No window
          x=x_last+dt*dxdt;
          end // No window
          if (window_type==1) begin // Jogelkar window
                    x=x_last+dt*dxdt*(1-pow((2*x_last/D-
                  1),(2*p_coeff)));
          end // Jogelkar window
          if (window_type==2) begin // Biolek window
                    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
          if (window_type==3) begin // Prodromakis window
                    x=x_last+dt*dxdt*J*(1-
                    pow((pow((x_last/D-0.5),2)+0.75),p_coeff));
          end  // Prodromakis window
          if (window_type==4) begin //Kvatinsky window
                    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);
          if  ((window_type==0)  ||  (window_type==4))  begin
          // No window
            w=w_last+dt*dwdt;
          end // No window
          if (window_type==1) begin // Jogelkar window
            w=w_last+dt*dwdt*(1-pow((2*w_last-1)  ,  (2  *
            p_coeff)));
          end // Jogelkar window
          if (window_type==2) begin // Biolek window
                    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
          if (window_type==3) begin // Prodromakis window
                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))<treshhold_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
```

V. Conclusions

A Verilog-A code that contains several models, useful for design in memristor-based circuits, is presented in this paper, as well as relevant window functions. This Verilog-A model can be used by circuit designers, since it is easy to use, contains several mathematical models, the parameters of already existing models can be easily changed, as well as additional mathematical models can be added.

Acknowledgment

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] L.O. Chua and S.M. Kang, “Memristive Devices and Systems,” *Proceedings of the IEEE*, Vol. 64, No. 2, pp. 209-223, February 1976.

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

[4] http://memristor.shorturl.com

[5] D. B. Strukov, G. S.Snider, D. R. Stewart, and R. S. Williams, "The Missing Memristor Found,” *Nature*, Vol. 453, pp. 80-83, May 2008.

[6] E. Lehtonen and M. Laiho, "CNN Using Memristors for Neighborhood Connections," *Proceedings of the International Workshop on Cellular Nanoscale Networks and their Applications*, pp. 1-4, February 2010.

[7] M. D. Pickett, D. B. Strukov, J. L. Borghetti, J. J. Yang, G. S. Snider, D. R. Stewart, and R. S. Williams, "Switching Dynamics in Titanium Dioxide Memristive Devices," *Journal of Applied Physics*, Vol. 106, No. 7, pp. 1-6, October 2009.

[8] Y. N. Joglekar and S. J. Wolf, “The Elusive Memristor: Properties of Basic Electrical Circuits,” *European Journal of Physics*, Vol. 30, No. 4, pp. 661-675, July 2009.

[9] Z. Biolek, D. Biolek, and V. Biolkova, "SPICE Model of Memristor with Nonlinear Dopant Drift," *Radioengineering*, Vol. 18, No. 2, Part 2, pp. 210-214, June 2009.

[10] T. Prodromakis, B. P. Peh, C. Papavassiliou, and C. Toumazou, “A Versatile Memristor Model with Non-linear Dopant Kinetics,” *IEEE Transactions on Electron Devices*, Vol. 58, No. 9, pp. 3099-3105, September 2011.

Table 1. The Different Memristor Models (Further description in [2])

Model	Linear ion drift [5]	Nonlinear ion drift [6]	Simmons tunneling barrier [7]	TEAM [3]
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
Threshold	None	None	None	i_{on}, i_{off}

Table 2. Comparison of Different Window Functions

Function	Jogelkar [8]	Biolek [9]	Prodromakis [10]	TEAM [3]
$f(x)/f(w)$	$f(w) = 1-(2w/D-1)^{2p}$	$f(w) = 1-(w/D-stp(-i))^{2p}$	$f(w)=j(1-[(w-0.5)^2+0.75]^p)$	$f_{on,off}=\exp[-\exp(\lvert x-x_{on,off}\rvert/w_c)]$
Fits memristor model	Linear/nonlinear ion drift/TEAM	Linear/nonlinear ion drift/TEAM	Linear/nonlinear ion drift/TEAM	TEAM for Simmons tunneling barrier fitting