Parametrization of Resistive Crossbar Arrays for Vector Matrix Multiplication

[Work in Progress]

Joachim Haase[1]
Fraunhofer IIS/EAS
Zeunerstr. 38
01069 Dresden, Germany

Abstract

Vector Matrix Multiplication (VMM) is a fundamental operation in machine learning algorithms focused on artificial neural networks and also many simulation codes. Implementations based on crossbar arrays provide a promising approach to perform this operation with an analogue circuit. In comparison to purely digital solutions, significant improvements in processing speed and power consumption can be expected when applying this approach. However, securing the accuracy is more difficult than in the digital case. Primary reasons include nonlinearities of essential resistive elements and non-zero resistances of wiring lines. Many publications have dealt with this topic in the recent years analysing the different influences in different ways. We provide a unified approach based on the well-known indefinite admittance matrix concept for the description of the terminal behaviour of analogue multi-poles for the parametrization of crossbar arrays and for the estimation of computational error limits. This paper describes work in progress. It illustrates the procedures through a number of examples using modelling and simulation capabilities of VHDL-AMS. This behavioural modelling language seems particularly suitable for investigations on tailored implementations using VMM. It combines the support of analogue mixed modelling and simulation with the facility to generate scalable architectures. Aspects of solving this task with Modelica are also discussed. Furthermore, it is also shown how symbolic methods might be used to consider resistances of wiring lines in the parametrization of crossbar arrays.

CCS Concepts

• Computing methodologies → Model development and analysis

Keywords

Vector Matrix Multiplication (VMM); resistive crossbar array; VHDL-AMS; Modelica.

1. Introduction

Vector Matrix Multiplications (VMM) are an essential task in deep learning algorithms. Their realization is a bottleneck in hardware implementations of dedicated machine learning solutions. The application of digital circuits for this purpose often increases processing time and power consumption beyond acceptable timing properties and low-power requirements. Therefore, alternatives have been investigated for some years. A promising solution for VMM is the usage of crossbar arrays with resistive elements [1-4] shown in Fig. 1. These analogue circuitries are proposed to be used as co-processing units in a digital environment.

In this way, hybrid computing from the end of the 1960s seems to be returning to the semiconductor era. However, this also raises challenges typical for analogue design, such as nonlinearities.

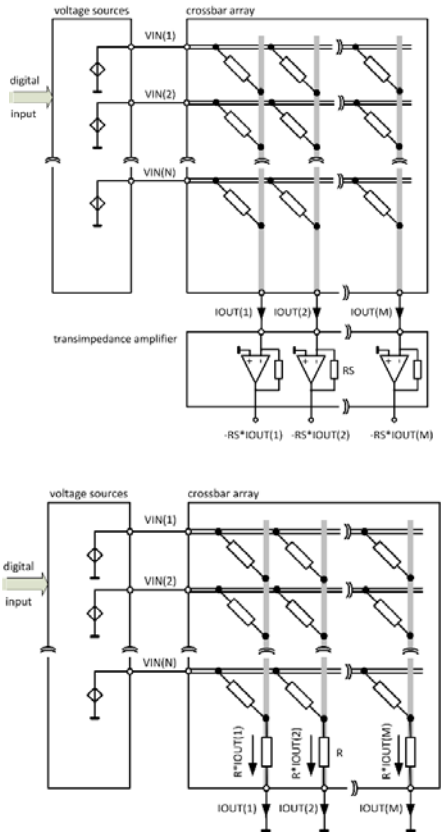

Fig. 1. VMM implementations with virtual ground (above) and terminal resistances (below)

[1] The author was with Fraunhofer Institute for Integrated Circuits (IIS), Division Engineering of Adaptive Systems (EAS) when starting this work. He is now reachable via e-mail joachim.haase@gmx.de

Fig. 1 shows two typical approaches for the analogue implementation of VMM. In both cases, a vector of *n* input voltages VIN is transformed into a vector of *m* currents IOUT using a crossbar arrangement. Horizontal and vertical crossbar lines are in different levels. The resistive elements that connect rows (horizontal lines) and columns (vertical lines) must be parametrized considering a given (positive real-valued) $m\ x\ n$ matrix *G* in such a way that $IOUT \sim G \cdot VIN$. The VMM implementation with virtual ground at the outputs of the crossbar array is also known as Dot-Product Engine (DPE) [4]. The outputs of the crossbar array are connected with inputs of transimpedance amplifiers (TIA) that convert the output currents into proportional voltages. In the case of the VMM implementation with terminal resistances the outputs of the crossbar array are directly connected to ground. The output currents flow through resistances *R* and the voltages over these resistances can be measured and further processed in the following circuit. Therefore in both cases, the VMM implementation can be embedded using Digital Analog and Analog Digital Converters (DAC and ADC, resp.) in a digital environment. Details considering current and future hardware implementations are explained in [1].

Several solutions for parametrization of crossbar arrangements were proposed in the past. A procedure that starts from an ideal crossbar array and tunes the cross-point resistive elements in order to get the expected behaviour for a circuit considering realistic constitutive relations of components is described for instance in [4]. Another iterative calibration procedure that systematically compensates the voltage drop related to finite wire resistances can be found in [5]. A special framework for the design of crossbar array structures is proposed in [6].

In this article we propose an approach that is based on the indefinite admittance matrix characterization of the ideal crossbar array. One purpose of this paper is to propose a straight-forward solution that allows to improve the accuracy of the parametrized arrays, estimate the error bounds and handle nonlinearities of junction elements. We parameterize junction components by minimizing the difference between the indefinite admittance matrix characterization of ideal and realistic crossbar arrangements. This procedure is straight forward, gives us an estimation of error bounds and a deeper insight into handling nonlinearities. The evaluation of the accuracy of VMM hardware implementations is essential for the algorithms that make use of them. Equation-based object-oriented (EOO) languages as VHDL-AMS [7] and Modelica [13] can help to investigate problems in this area.

We assume that the given matrix *G* consists only of real positive elements that can be realized within the limits of a given technology. Otherwise an upstream step is required to rewrite the original task. We also assume that it is not necessary to consider the RC delay. Thus, we restrict our presentation to the resistive behaviour. For further reading, we refer for instance to [3]. Furthermore, we do not take noise aspect into consideration. It is not figured out how the values of the resistive junction elements can be set.

2. Indefinite admittance matrix representation of crossbar array

The concept of the indefinite admittance matrix representation for the description of a linear multi-pole goes back to Shekal [8] and can also be transferred in a modified form to the description of the terminal behaviour of nonlinear multi-poles [9]. This concept can be applied to the analysis and parametrization of crossbar arrays for VMM.

Fig. 2 sketches the approach for the admittance description of a crossbar array used for VMM. In the upper part, the general situation is described where the voltage and current values *VIN(i), VOUT(j), IN(i)* and *IOUT(j)* with *i* in *{1, …, n}* and *j* in *{1, …, m}* are real values and $Y_{11}, Y_{12}, Y_{21}, Y_{22}$ are real-valued $n\ x\ n$, $n\ x\ m$, $m\ x\ n$ and $m\ x\ m$ matrices, resp.

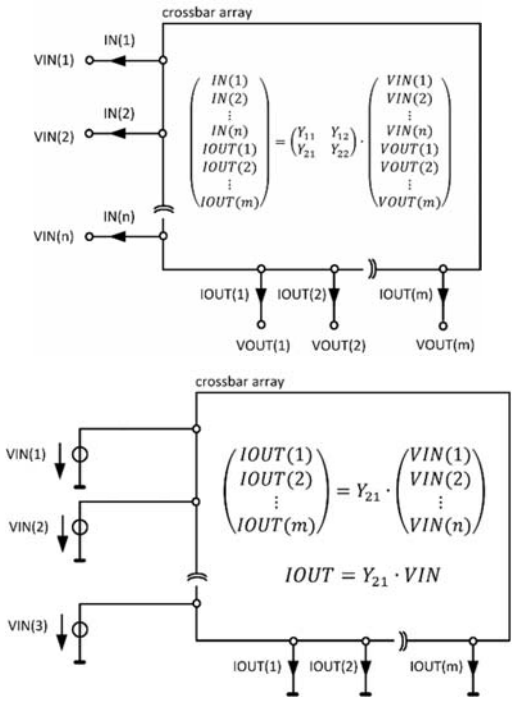

Fig. 2. Admittance descriptions of the crossbar array

The lower part figures out the special situation related to VMM as shown in Fig.1. The outputs are connected to virtual ground or ground what means that all output voltage *VOUT(j)* are zero and the input currents *IIN(i)* are not interesting. We can now realize a Vector Matrix Operation with a real-valued matrix *G* if Y_{21} is equal (or proportional) to the $m\ x\ n$ matrix *G*. This means that

$$\begin{pmatrix} IOUT(1) \\ IOUT(2) \\ \vdots \\ IOUT(m) \end{pmatrix} = G \cdot \begin{pmatrix} VIN(1) \\ VIN(2) \\ \vdots \\ VIN(n) \end{pmatrix} = \begin{pmatrix} G_{11} & G_{12} & \cdots & G_{1n} \\ G_{21} & G_{22} & \cdots & G_{2n} \\ \vdots & & \ddots & \vdots \\ G_{m1} & G_{m2} & \cdots & G_{mn} \end{pmatrix} \cdot \begin{pmatrix} VIN(1) \\ VIN(2) \\ \vdots \\ VIN\ (n) \end{pmatrix} \quad (1)$$

If instead of *G* we can only implement a matrix $\bar{G}$ with a special crossbar array we get the currents $\overline{IOUT} = \bar{G} \cdot VIN$. Then, the error can be measured by $\|IOUT - \overline{IOUT}\| = \|(G - \bar{G}) \cdot VIN\| \leq \|G - \bar{G}\| \cdot \|VIN\|$. Therefore, in order to reduce the VMM error we have to minimize $\|G - \bar{G}\|$ in the operating area of the crossbar array. If we use as vector norm the maximum norm then the corresponding matrix norm is the row-sum norm. That means $\|G - \bar{G}\|_\infty = \max_{i=1,\ldots,m} \sum_j^n |G_{ij} - \bar{G}_{ij}|$.

3. Crossbar array with linear and nonlinear junction elements

If the matrix *G* consists only of real positive elements the Dot-Product Engine can be implemented in the ideal case with a crossbar array with zero resistances of wiring lines and resistive junction elements that can be described by constant linear admittances as shown in the upper part of Fig. 3 and described in [4].

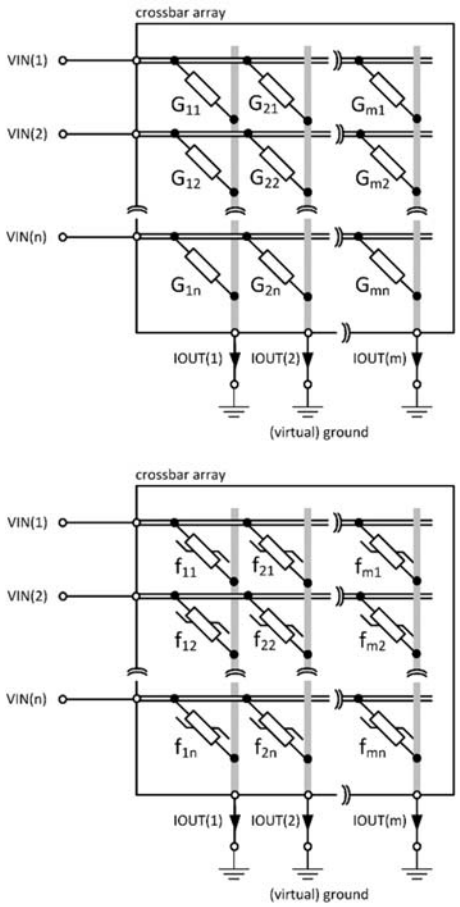

Fig. 3. Assignment of matrix *G* elements to linear junction admittances in the ideal case (above) and with nonlinear elements (below)

Let us now use nonlinear junction elements instead of linear admittances. That means, we use two-terminal components with a constitutive relation described by $I = f_{ij}(V)$ where *I* and *V* are the branch current and voltage, resp. and f_{ij} is a real-valued function $f_{ij}: \mathbb{R} \to \mathbb{R}$. The corresponding crossbar array is sketched in the lower part of Fig. 3. It must be possible to modify the functions f_{ij} in such a way that they approximate the admittances G_{ij}. It is usually proposed to do this in such a way that [3]

$$G_{required} = \lim_{V \to 0} \frac{f_{Nonlinear}(V)}{V} = \left. \frac{df_{Nonlinear}(V)}{dV} \right|_{V=0} \tag{2}$$

where the real-valued setpoint is $G_{required}$. For small branch voltages *V*, this seems to be an acceptable solution. However, the objective should be to minimize the maximum deviation of the junction elements' admittances from $G_{required}$ as a result of the discussion at the end of the previous section.

To answer the question how the nonlinear admittances should be parametrized we recall the generalization of the substitution theorem [10]. For given *VIN* voltages we can determine branch voltages and currents of a crossbar array with nonlinear elements.

Let $\overline{V_{ij}}$ and $I_{ij} = f_{ij}(\overline{V_{ij}})$ be branch voltage and branch current of a two-terminal nonlinear junction element at a solution. Without loss of generality, we can assume that there is only one solution point for the network analysis task. Then we can replace the nonlinear element with a linear admittance with value $\overline{G_{ij}} = \frac{\overline{I_{ij}}}{\overline{V_{ij}}}$ and will get the same solution for the new network analysis task. Let G_{ij} be the required admittance in the ideal case then the error of the vector matrix multiplication depends on $\left|G_{ij} - \overline{G_{ij}}\right|$.

This leads us to a proposal for the parametrization of the functions *f*. We assume that the branch voltage of a two-terminal junction element belongs to an interval $[V_{min}, V_{max}]$. Then we have to parametrize the function f_{ij} of a two-terminal junction element in such a way that considering all possibly $\overline{G_{ij}}$ the maximum of all possibly differences $\left|G_{ij} - \overline{G_{ij}}\right|$ is as small as possible. Therefore, we make the subsequent

PROPOSAL 1. In the ideal case the linear admittance of a two-terminal junction element shall be the real value $G_{required}$. The constitutive relation of the nonlinear two-terminal junction element shall be described by a function $f: \mathbb{R} \to \mathbb{R}, V \mapsto f(V)$. Assuming that the expected branch voltage belongs at solution points to the interval $[V_{min}, V_{max}]$, then the function *f* should be adapted to $f_{Adapted}$ in such a way that

$$\max_{V \in [V_{min}, V_{max}]} \left| G_{required} - \frac{f_{Adapted}(V)}{V} \right| \to min \tag{3}$$

This is in general an optimization problem. We mention that (3) is equivalent to

$$\max_{V \in [V_{min}, V_{max}]} \left| \frac{G_{required} \cdot V - f_{Adapted}(V)}{G_{required} \cdot V} \right| \to min \tag{4}$$

This means using $f_{Adapted}$ for the constitutive relation of nonlinear junction element, the relative error of the current through the junction element compared to the ideal case is minimized. In a similar way, we could require to minimize the absolute error.

Example

We demonstrate the procedure with the help of a simple example. The characteristic of the nonlinear two-terminal junction elements shall be given by $I = I_0 \cdot \exp\left(-\frac{d}{d_0}\right) \cdot sinh\left(\frac{V}{V_0}\right)$ with $I_0 = 10^{-3}, d_0 = 0.25 \cdot 10^{-9}$ and $V_0 = 0.25$ [3, 11]. The functions can be adapted by changing the average tunneling gap distance *d*. The branch voltages of junction elements in the crossbar array shall be between $V_{min} = 0.0$ and $V_{max} = 0.3$. Thus, for a given $G_{required} < I_0/V_0$ we can determine based on (2)

$$d_{Nonlinear} = -d_0 \cdot \left(ln(G_{required}) - ln\left(\frac{I_0}{V_0}\right) \right) \tag{5}$$

Instead of solving the optimization problem (3) directly, we got a (possibly suboptimal) solution by setting $G_{required}$ to the

arithmetic mean of the minimum and maximum replacement admittances of the nonlinear junction element for $V = V_{min} = 0$ and $V = V_{max}$, resp. We determined

$$d_{Adapted} = -d_0 \cdot \left(ln\left(2 \cdot G_{required}\right) - ln\left(\frac{I_0}{V_0} + I_0 \cdot \frac{sinh\frac{V_{max}}{V_0}}{V_{max}}\right)\right) \quad (6)$$

Fig. 4 shows $G_{required}$ and replacement admittances based on (5) and (6).

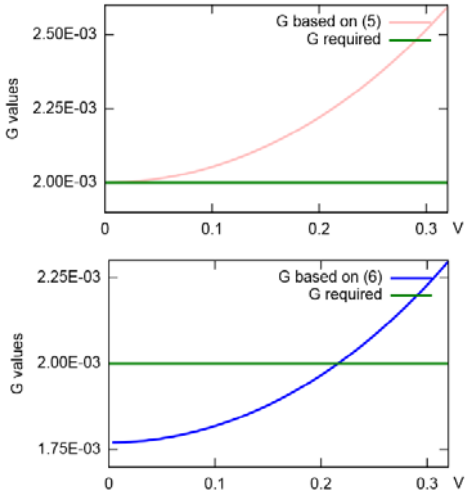

Fig. 4. $G_{required} = 2 \cdot 10^{-3}$ and replacement admittances based on (5) (above) and (6) (below) for *V* between 0.0 and 0.3

We demonstrate the consequences of the presented approach by simulating implementations based on the upper part of Fig. 1 (crossbar array with TIAs) with $R_S = 10^3$ and for the $m\,x\,n$ matrix $G = 10^{-3} \cdot \begin{pmatrix} 2.0 & 0.8 \\ 0.5 & 0.9 \\ 0.6 & 0.7 \end{pmatrix}$. An improvement with nonlinear elements based on (6) instead (5) can be especially shown for higher voltages *VIN*.

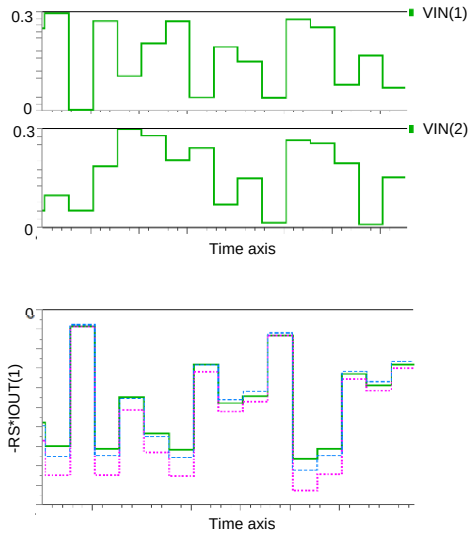

Fig. 5. Input voltages VIN(1) and VIN(2) (above) and -RS*IOUT(1) (below) for ideal case (green solid) and using $f_{Nonlinear}$ (pink dotted) and $f_{Adatped}$ (blue dashed)

Fig. 5 shows $-RS \cdot IOUT(1)$ for implementations of the crossbar array with ideal and nonlinear junction elements □

4. Considering wiring resistances of lines in crossbar arrays

We are now interested in the influence of (linear) wiring resistances and possibly terminal resistances in crossbar array parametrization. Below, we will present methods to parametrize linear junction elements using symbolic and numerical approaches.

PROPOSAL 2. Assuming that the requirements for currents sums and constitutive relations of possibly existing terminal resistances of a crossbar array with linear junction admittances considering wiring resistances of lines can be written in the form

$(A\ B\ C) \cdot (IOUT\ VI\ VIN)^T = \vec{0}$ where *IOUT*, *VI* and *VIN* are real-valued *m*-, *r*- and *n*-dimensional vectors, resp. and *A*, *B*, and *C* are real matrices of dimensions $(m + r)\,x\,m$, $(m + r)\,x\,r$ and $(m + r)\,x\,n$, resp. *VI* are node voltages of *r* internal nodes that have to be taken into account. The matrices are defined using fixed values and variables to characterize wire resistances and admittances of junction elements. If $\begin{pmatrix} I_m & \mathbf{0} & D \\ \mathbf{0} & I_r & E \end{pmatrix}$ is the reduced row echelon form of $(A\ B\ C)$ then $IOUT = -D \cdot VIN$ describes the terminal behaviour of the crossbar array. I_m and I_r are unity matrices. *D* is a $m\,x\,n$ matrix. Comparing *–D* with a given $m\,x\,n$ matrix *G* for Vector Matrix Multiplication parameters of the crossbar array can be parametrized.

Example

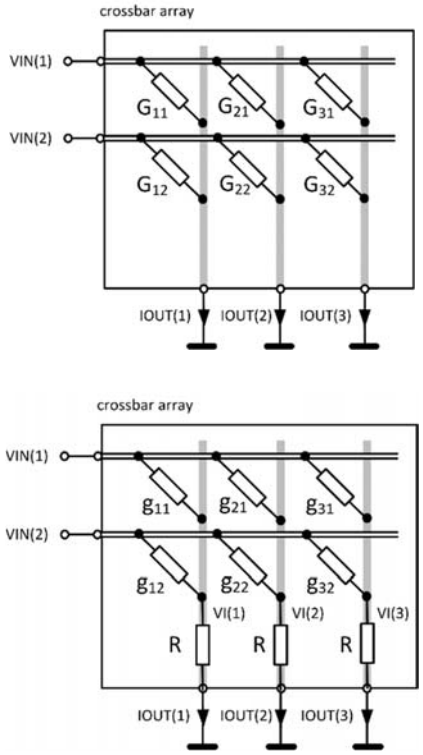

Fig. 6. Ideal crossbar array for VMM with matrix *G* (above) and crossbar array with terminal resistances (below)

For the crossbar array with terminal resistances in Fig. 6 we can establish (A B C) and get the reduced row echelon form using a symbolic math program as follows

$$(A\ B\ C) = \begin{pmatrix} 1 & 0 & 0 & g_{11}+g_{12} & 0 & 0 & -g_{11} & -g_{12} \\ 0 & 1 & 0 & 0 & g_{21}+g_{22} & 0 & -g_{21} & -g_{22} \\ 0 & 0 & 1 & 0 & 0 & g_{31}+g_{32} & -g_{31} & -g_{32} \\ 1 & 0 & 0 & -\frac{1}{R} & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -\frac{1}{R} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -\frac{1}{R} & 0 & 0 \end{pmatrix}$$

$$\Rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & -\frac{g_{11}}{Rg_{11}+Rg_{12}+1} & -\frac{g_{12}}{Rg_{11}+Rg_{12}+1} \\ 0 & 1 & 0 & 0 & 0 & 0 & -\frac{g_{21}}{Rg_{21}+Rg_{22}+1} & -\frac{g_{22}}{Rg_{21}+Rg_{22}+1} \\ 0 & 0 & 1 & 0 & 0 & 0 & -\frac{g_{31}}{Rg_{31}+Rg_{32}+1} & -\frac{g_{32}}{Rg_{31}+Rg_{32}+1} \\ 0 & 0 & 0 & 1 & 0 & 0 & -\frac{Rg_{11}}{Rg_{11}+Rg_{12}+1} & -\frac{Rg_{12}}{Rg_{11}+Rg_{12}+1} \\ 0 & 0 & 0 & 0 & 1 & 0 & -\frac{Rg_{21}}{Rg_{21}+Rg_{22}+1} & -\frac{Rg_{22}}{Rg_{21}+Rg_{22}+1} \\ 0 & 0 & 0 & 0 & 0 & 1 & -\frac{Rg_{31}}{Rg_{31}+Rg_{32}+1} & -\frac{Rg_{32}}{Rg_{31}+Rg_{32}+1} \end{pmatrix}$$

For small examples for instance, the function `row reduce` of WolframAlpha can be used for this purpose [12]. Setting $G_{11} = \frac{g_{11}}{R\cdot g_{11}+R\cdot g_{12}+1}, G_{12} = \frac{g_{12}}{R\cdot g_{11}+R\cdot g_{12}+1}$ etc. we get for $G = 10^{-3}\cdot \begin{pmatrix} 2.0 & 0.8 \\ 0.5 & 0.9 \\ 0.6 & 0.7 \end{pmatrix}$ and *R=300* the admittances $g_{11} = \frac{1}{80} = 12.5\cdot 10^{-3}, g_{12} = \frac{1}{200} = 5\cdot 10^{-3}, g_{21} = \frac{1}{1160} \approx 0.862\cdot 10^{-3}$ and $g_{22} = \frac{9}{5800} \approx 1.551\cdot 10^{-3}, g_{31} = \frac{3}{3050} \approx 0.983\cdot 10^{-3}, g_{32} = \frac{7}{6100} \approx 1.147\cdot 10^{-3}$. Already in this simple example, we see a great difference between values of junction admittances in the ideal case and the implementation with terminal resistances. We can also observe that for greater values of *R* we do not get a solution with only positive values of the g_{ij} □

Wiring resistances in crossbar arrays in semiconductor technologies depend on the feature size of a technology node. In [4], wiring resistances of 10 Ω and input/output resistances of 100 Ω are assumed. Lee at al. [5] report values of about 4.53 Ω, 2.97 Ω and 1.55 Ω for the resistances between adjacent cells in 16 nm, 22 nm and 32 nm technology nodes, resp. These values also have to be taken into consideration during the parametrization of the junction elements of crossbar arrays. If initial values of the admittances of the junction elements are given then their values in an array with wiring resistances can be determined in an iterative process. For such an array we can determine the admittances (see Fig. 2) with *n* network simulation tasks. Therefore we have to stimulate with *VIN* voltages whereby successively one input is excited with value 1 and all remaining voltage are set to zero. The *IOUT* currents deliver the results. This procedure defines a function $\tilde{G}\colon \mathbb{R}^{n\cdot m} \longrightarrow \mathbb{R}^{nxm}, g \mapsto \tilde{G}(g)$ where g is a vector of the values of the admittances of the junction elements in a crossbar array considering wiring resistances.

PROPOSAL 3. Let $\tilde{G}\colon \mathbb{R}^{n\cdot m} \longrightarrow \mathbb{R}^{nxm}$ be a function that determines the admittance description Y_{21} of the arrangement shown in the lower part of Fig. 2. Let *G* be the required matrix for Vector Matrix Multiplication. Then the crossbar array has to be parametrized with junction elements g^* that fulfill the matrix equation element by element $\tilde{G}(g^*) - G = \mathbf{0}$. This task can be performed by a solver for non-linear equation systems.

Several iterative procedures have been proposed for parametrization of crossbar arrays in the past (see for instance [3, 4]). However, it seems that the presented way helps to give a better understanding of the solution steps. Further efforts are necessary to combine the suggestions given by proposals 1 to 3.

5. VHDL-AMS and Modelica for crossbar array modelling and simulation

We can use VHDL-AMS [7] to develop models to check the procedures described. Using this behavioural modelling language it is possible to describe scalable network topologies. This is especially supported by the generate statement. Fig. 7 demonstrates the description of an ideal crossbar array as shown in Fig. 1 with any number `N` of columns and `M` of rows.

```
library IEEE, ANEU;
use ANEU.ARRAY_CONFIGURATION.all;
use IEEE.ELECTRICAL_SYSTEMS.all;

entity CROSSBAR_ARRAY is
 generic (G_VALUES : REAL_VECTOR
          (1 to NUMBER_OF_NODES)   -- M*N
          := (others => 0.0));

 port(terminal E_INPUT  :
              ELECTRICAL_VECTOR (1 to N);
      terminal E_OUTPUT :
              ELECTRICAL_VECTOR (1 to M));
end entity CROSSBAR_ARRAY;

architecture IDEAL of CROSSBAR_ARRAY is
begin
ROW: for I in 1 to N generate
COL:  for J in 1 to M generate
      begin
        G: entity ADMITTANCE(A0)
           generic map
             (G => G_VALUES (I + (J-1)*N))
           port map
             (P => E_INPUT(I),
              N => E_OUTPUT(J));
      end generate COL;
     end generate ROW;
end architecture IDEAL;
```

Fig. 7. VHDL-AMS model of an ideal crossbar array

The constants `N` and `M` are provided by the package CONFIGURATION. The elements G_{ji} of the ideal matrix *G* used in equation (1) are given by `G_VALUES (I+(J-1*N)`. The mapping of the matrix to a vector avoids in VHDL-AMS a user-defined declaration of a two-dimensional real-valued array. The architecture `A0` of `ADMITTANCE` belongs to the working library `ANEU`. The generate statements assign to each junction admittance as parameter the corresponding element of the matrix *G* and the connection nodes. Connection nodes are in the case of the ideal crossbar array the input and output terminals of the crossbar array. The interface of the `ADMITTANCE` model is given by

```
entity ADMITTANCE
    generic (G : REAL := 0.0; -- admittance value
             T : REAL := 293.15);  -- Temperature
    port (terminal P, N : ELECTRICAL);
end entity ADMITTANCE;
```

A branch with branch voltage V and branch current I connects `P` and `N`. The VHDL-AMS model can take into consideration the power spectral density of the branch noise current at the quiescent domain solution point (see [7], section 14.9). With the Boltzmann constant *k* the constitutive relation of the noisy admittance can be described by $I = G\cdot V + 4kTG$. The Modelica [13] model equivalent to Fig. 7 uses `Modelica.Electrical.Analog.Basic.Conductor` as model for the junction admittances. However, small signal noise analysis is not supported in Modelica.

```
model crossbar_array_ideal
  "Ideal crossbar array"
  parameter Integer n = 1;
  parameter Integer m = 1;
  parameter Real    g_values[m*n]
  "Values of matrix coefficients";

  Pin e_input [n] "Electrical input pins";
  Pin e_output[m] "Electrical output pins";

  Conductor cond[m*n]
  (G = {g_values[i] for i in 1:m*n})
  "Junction admittances";

  SI.Power LossPower "Total loss power";
algorithm
 LossPower := 0.0;
 for k in 1:m*n loop
  LossPower := cond[k].LossPower + LossPower;
 end for;
equation
 for i in 1:n loop
  for j in 1:m loop
     connect (cond[i+(j-1)*n].p,e_input[i]);
     connect (cond[i+(j-1)*n].n,e_output[j]);
  end for;
 end for;
end crossbar_array_ideal;
```

Fig. 8. Modelica model of an ideal crossbar array

Fig. 8 shows the Modelica model of the ideal crossbar array equivalent to the VHLD-AMS model (see Fig. 7). It accesses the instructions

```
import Modelica.Electrical.Analog.Basic.Conductor;
import Modelica.Electrical.Analog.Interfaces.Pin;
import SI = Modelica.SIunits;
```

The model parameters `n` and `m` determine the lengths of input and output pin vectors `e_input` and `e_output` resp. One part of the functionality of the VHDL-AMS `generate` statement is provided by the support of the declaration of a vector `cond` of `Conductor` components. The model parameters `n` and `m` can be used for this declaration and for the assignment of the `g_values` to the components applying the rules for the modification of array elements (see [13], section 7.2.5). The other part of the `generate` functionality is carried out in the `equation` section. The `connect` statements describe the topology of the crossbar array. This is done in the VHDL-AMS model using the `port maps`.

Modelica also allows "access members of a class instance using dot notation" (see [13], section 3.6.6). This property can be used to sum up the loss powers of the components as applied in the `algorithm` section of the model. A similar feature is not provided in VHDL-AMS where a work-around has to be installed to handle this problem.

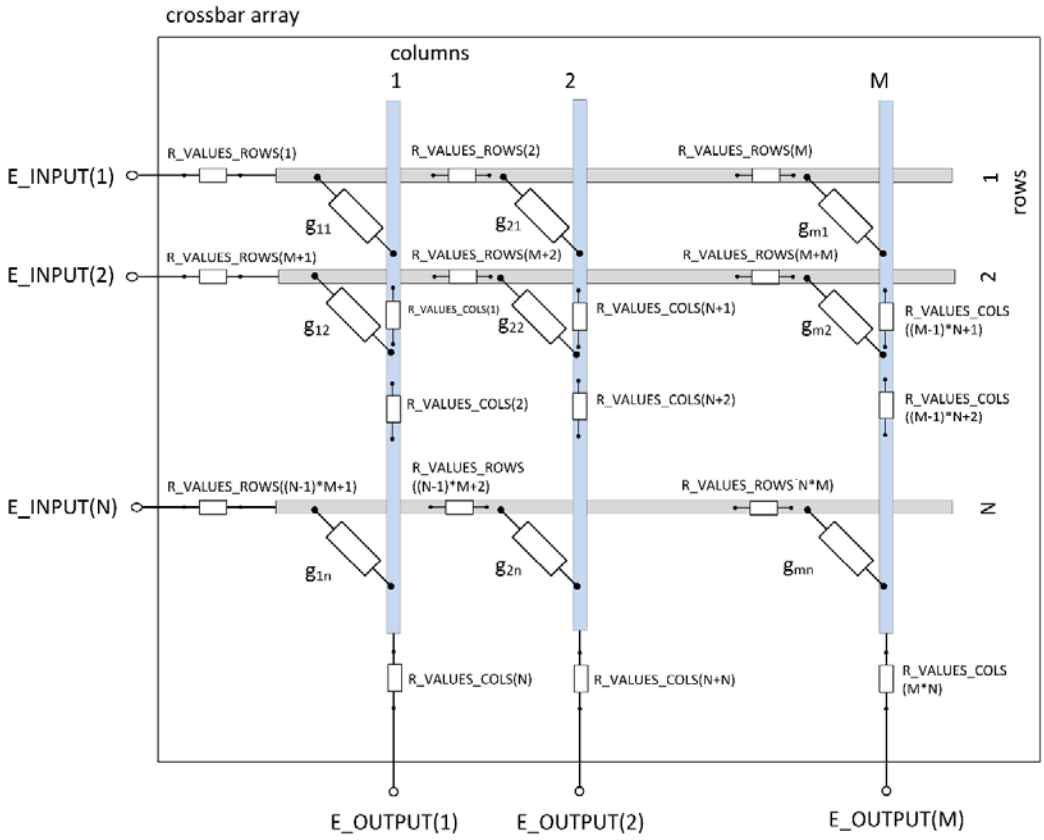

Fig. 9. Schematic of a general crossbar array that includes resistances of wiring lines

Fig. 9 shows the general case that was discussed in section 4. The VHDL-AMS model of this arrangement uses models for single rows and columns. The principal approach for these models is demonstrated in Fig. 10.

```
library ANEU, IEEE;
use ANEU.ARRAY_CONFIGURATION.all;
use IEEE.ELECTRICAL_SYSTEMS.all;

entity CROSSBAR_ARRAY_ROW is

 generic (R_VALUES : REAL_VECTOR (1 to M)
                   := (others => 0.0));

 port (terminal E_IN  : ELECTRICAL;
       terminal E_ROW :
                ELECTRICAL_VECTOR(1 to M));
end entity CROSSBAR_ARRAY_ROW;

architecture A0 of CROSSBAR_ARRAY_ROW is
begin
R_IN:  entity RESISTOR(A0)
       generic map (R => R_VALUES(1))
       port map    (P => E_IN, N => E_ROW(1));

R_ROW: for I in 2 to M generate
        RI: entity RESISTOR (A0)
             generic map (R => R_VALUES(I))
             port map
                (P => E_ROW(I-1),
                 N => E_ROW(I));
       end generate R_ROW;
end architecture A0;
```

Fig. 10. VHDL-AMS model of a row for a general crossbar array

N row models, M column models and M*N (=NUMBER_OF_NODES) models of the junction elements built up the model for the general crossbar array that includes wiring resistances of lines as shown in Fig. 8.

```
library ANEU, IEEE;
use ANEU.ARRAY_CONFIGURATION.all;
use IEEE.ELECTRICAL_SYSTEMS.all;

entity CROSSBAR_ARRAY is
 generic (G_VALUES : REAL_VECTOR
          (1 to NUMBER_OF_NODES)   -- M*N
          := (others => 0.0));
          R_VALUES_ROWS : REAL_VECTOR
          (1 to NUMBER_OF_NODES)
          := (others => 0.0);
          R_VALUES_COLS :
          REAL_VECTOR (1 to NUMBER_OF_NODES)
          := (others => 0.0));
port (terminal E_INPUT  :
               ELECTRICAL_VECTOR (1 to N);
      terminal E_OUTPUT :
               ELECTRICAL_VECTOR (1 to M));
end entity CROSSBAR_ARRAY;

architecture WITH_ROW_COLUMN_RESISTORS of
CROSSBAR_ARRAY is
 terminal ROW_CONNECTIONS :
       ELECTRICAL_VECTOR (1 to NUMBER_OF_NODES);
       -- connection points in row lines
 terminal COL_CONNECTIONS :
       ELECTRICAL_VECTOR (1 to NUMBER_OF_NODES);
       -- connection points in column lines
begin
-- Connection of E_INPUT terminals with rows
GEN_ROWS:for AI in 1 to N generate
  constant IDX_START : POSITIVE := (AI-1)*M + 1;
  constant IDX_END   : POSITIVE := (AI-1)*M + M;
  constant R_ROW_AI  : REAL_VECTOR (1 to M) :=
           R_VALUES_ROWS(IDX_START to IDX_END);
  begin
     ROW : entity CROSSBAR_ARRAY_ROW(A0)
            generic map (R_VALUES => R_ROW_AI)
            port map (E_IN  => E_INPUT(AI),
                      E_ROW => ROW_CONNECTIONS
                      (IDX_START to IDX_END));
end generate GEN_ROWS;

-- Connection of E_OUTPUT terminals with columns
GEN_COLUMNS:for AJ in 1 to M generate
  constant IDX_START : POSITIVE := (AJ-1)*N + 1;
  constant IDX_END   : POSITIVE := (AJ-1)*N + N;
  constant R_COL_AJ  : REAL_VECTOR (1 to N) :=
           R_VALUES_COLS (IDX_START to IDX_END);
  begin
    COL : entity CROSSBAR_ARRAY_COLUMN(A0)
          generic map (R_VALUES => R_COL_AJ)
          port map (E_OUT => E_OUTPUT(AJ),
                    E_COLUMN => COL_CONNECTIONS
                    (IDX_START to IDX_END));
end generate GEN_COLUMNS;
-- Connection of junction admittances
G_ROW: for I in 1 to N generate  -- index row
G_COL: for J in 1 to M generate  -- index column
        constant IDX_TERMINAL_IN_ROW :
                 POSITIVE := (I-1)*M + J;
        constant IDX_TERMINAL_IN_COL :
                 POSITIVE := (J-1)*N + I;
        begin
         G1: entity ADMITTANCE(A0)
               generic map(G =>
                           G_VALUES(I+(J-1)*N))
               port map (P => ROW_CONNECTIONS
                          (IDX_TERMINAL_IN_ROW),
                         N => COL_CONNECTIONS
                          (IDX_TERMINAL_IN_COL)
                         );
        end generate G_COL;
     end generate G_ROW;
end architecture WITH_ROW_COLUMN_RESISTORS;
```

Fig. 11. VHDL-AMS model of the general crossbar array shown in Fig. 9 with linear admittances as junction elements

This model can be used to check several approaches for the parametrization of crossbar array models by simulation. It can be investigated whether the accuracy of hardware realizations of VMM is as high as necessary. Instead of linear admittances, other components can be used as junction elements as for instance the nonlinear elements described by the equations (5) and (6). In an electronic design flow that supports VHDL-AMS, the models can be integrated. The inputs of the crossbar array can be connected to a Digital Analog Converter (DAC). The outputs of the transimpedance amplifiers (see Fig. 1) can be transformed by ADCs to digital values. Furthermore, capacitors can be added to the crossbar array models in order to check whether the timing behaviour of the crossbar array is adapted to the sampling period of the digital signal processing units. However, RC delay for advanced semiconductor technologies is expected to be sub-ns and can be ignored [4].

In principle, Modelica has similar capabilities. Table 1 summarizes features of VHDL-AMS and Modelica to model crossbar arrays.

Table 1. Support of crossbar array modelling by VHDL-AMS and Modelica

	VHDL-AMS	Modelica
Scalable topologies	**`generate`** statement	vector/array of components declaration loops with **`connect`** statements
Nonlinear junction elements	supported	supported
Dynamic behaviour	supported	supported
Total power determination	implementation of special solutions required	access to loss power of instances with dot notation
Small signal noise analysis	noise source quantity	not supported

6. Conclusions

Vector Matrix Multiplication is a fundamental operation in machine learning algorithms. Approaches to replace the digital implementation by analogue circuits based on crossbar arrangements have been investigated since some years. It was reported about hardware implementation and simulation studies. In this paper, we tried to discuss problems related to the parametrization based on known results from the network theory. We gave proposals to handle nonlinearities of junction elements and the influence of wiring resistances based on symbolic and numerical methods and provided a unified approach based on the well-known indefinite admittance matrix concept. We illustrated the approach using simple examples. VHDL-AMS was used for simulation experiments. It provides the features that are necessary for first studies on new ideas considering crossbar arrays. It seems plausible that VHDL-AMS might also be of value for the investigation of synthesis problems for hardware solutions for other linear algebraic problems that are discussed in [14]. Modelica is another language suitable for these investigations. This paper can be only a first step. Further work has to be done to efficiently implement and test solutions based on the given proposals. There are limits of accuracy using crossbar arrangements for VMM. The question occurs for which task which inaccuracy can be accepted. An approach could be to include "inaccurate" models for VMM in the development of algorithms. It might be possible to use Modelica based features like the Functional Mock-up Interface (FMI) for this task.

7. Acknowledgements

The author thanks the reviewers for information on missing explanations and also the Fraunhofer-Gesellschaft for their support.

8. References

[1] W. Woods and C. Teuscher 2017. Approximate vector matrix multiplication implementations for neuromorphic applications using memristive crossbars. *2017 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, Newport, RI, 2017, pp. 103-108. DOI=http://dx.doi.org/10.1109/NANOARCH.2017.8053729

[2] H. Assaf, Y. Savaria, and M. Sawan 2018. Vector Matrix Multiplication Using Crossbar Arrays: A Comparative Analysis. *25th IEEE Int. Conference on Electronics, Circuits and Systems (ICECS)*, Bordeaux, 2018, pp. 609-612. DOI=http://dx.doi.org/10.1109/ICECS.2018.8617942

[3] Peng Gu et al. 2015. Technological exploration of RRAM crossbar array for matrix-vector multiplication. *The 20th Asia and South Pacific Design Automation Conference*, Chiba, 2015, pp. 106-111. DOI= http://dx.doi.org/10.1109/ASPDAC.2015.7058989

[4] M. Hu et al. 2016. Dot-product engine for neuromorphic computing: Programming 1T1M crossbar to accelerate matrix-vector multiplication. *53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, Austin, TX, 2016, pp. 1-6. DOI=http://dx.doi.org/10.1145/2897937.2898010

[5] Y. K. Lee et al. 2019. Matrix Mapping on Crossbar Memory Arrays with Resistive Interconnects and Its Use in In-Memory Compression of Biosignals. *Micromachines* 2019, 10(5), 306. DOI= http://dx.doi.org/10.3390/mi10050306

[6] S. Jain et al. 2019. RxNN: A Framework for Evaluating Deep Neural Networks on Resistive Crossbars. last revised 18 Jan 2019. arXiv.1809.00072v2 [cs.ET]. Available: https://arxiv.org/abs/1809.00072

[7] IEEE Standard VHDL Analog and Mixed-Signal Extensions 2017. In *IEEE Std 1076.1-2017 (Revision of IEEE Std 1076.1-2007)*, pp.1-672, 26 Jan. 2018. DOI=http://dx.doi.org/10.1109/IEEESTD.2018.8267464

[8] J. Shekel 1952. Matrix Representation of Transistor Circuits. In *Proceedings of the IRE*, vol. 40, no. 11, pp. 1493-1497, Nov. 1952. DOI=http://dx.doi.org/10.1109/JRPROC.1952.273987

[9] A. Reibiger 2013. Networks, multipoles and multiports. *2013 European Conference on Circuit Theory and Design (ECCTD)*, Dresden, 2013, pp. 1-37. DOI= http://dx.doi.org/10.1109/ECCTD.2013.6662191

[10] J. Haase 1985. On generalizations and applications of the substitution theorem. In *Proc. European Conference on Circuit Theory Design (ECCTD'85)*, B5.2, Prague, Sept. 2-6, 1985, pp. 220-223.

[11] Z. Jiang and H.-S. Philip Wong 2014. Stanford University Resistive-Switching Random Access Memory (RRAM) Verilog-A Model. nanoHUB, 2014. DOI= http://dx.doi.org/10.4231/D37H1DN48

[12] Wolfram Alpha. Available: https://www.wolframalpha.com/

[13] *Modelica® - A Unified Object Oriented Language for Systems Modeling: Language Reference Manual Version 3.4,* Modelica Association, April 10, 2017. Available: https://www.modelica.org/documents

[14] Zhong Sun et al. 2019. Solving matrix equations in one step with cross-point resistive arrays. In *Proc. of the National Academy of Sciences,* Mar 2019, 116 (10) 4123-4128. DOI=http://dx.doi.org/10.1073/pnas.1815682116