2013 10th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)
Mexico City, Mexico. September 30-October 4, 2013

FGPA Development and implementation of a solar panel emulator

G.M. Tornez-Xavier[1], F. Gómez-Castañeda[1], J.A. Moreno-Cadenas[1], L.M. Flores-Nava[1]
[1]Department of Electrical Engineering, CINVESTAV-IPN, Mexico D.F., Mexico
Phone (52) 55 5747 3800 Ext. 6261
E-mail: {gtornez, fgomez, jmoreno, lmflores} @cinvestav.mx

***Abstract* — This work describes the development and FPGA implementation of a solar panel emulator. First we created the electric analog model of the solar panel using the Mentor Graphics framework, using the irradiance and temperature variables of a meteorological database as input signals and then obtaining the short circuit current and the open circuit voltage parameters to finally train an artificial neural network using Matlab, to perform the modeling of the response of the solar panel. Once the neural network was optimized, this was described in VHDL to simulate its response, to finally make its implementation in FPGA digital device and to be able to compare these results with those of a commercial solar panel.**

***Keywords* — Photovoltaic module, simulation, artificial neural networks, VHDL, FPGA.**

I. INTRODUCTION

Today, there is a growing interest in the use of renewable energy, particularly solar energy due to its high availability around the world, letting alone the fact that it is a non-polluting energy, also there is a growing need of working and understanding photovoltaic systems. In this work, we will focus on the development of the solar panel emulator which as any photovoltaic panel, will do the conversion of solar energy into electric power whose key components are the solar cells, which can be interconnected in a serial, parallel or mixed way, depending on the application of interest. The estimation of an analytical model of a solar module might be difficult due to the large number of environmental factors involved; by using artificial neural networks (ANN's) greatly simplifies calculations allowing to emulate its behavior, the ANN's are presented within two categories: implementation in software which offers great flexibility in training and simulation of the network for general purpose applications and hardware implementation which will allow us to perform simulations in real time, in this work we will try to cover both aspects. The FGPA device belongs to the family of programmable devices (PLD's) this will allow us to quickly change and adapt any changes within the network architecture so its real time response can be experimented.

II. DATABASE

We obtained a meteorological database of the Mexican state of Nayarit, containing information of 1097 days corresponding to a period of three years, from which the irradiance (H) and temperature (T) values were obtained and analyzed daily from that Mexican state. The temperature range is ($11.8^{\circ}C < T < 39.6^{\circ}C$) and the range of the irradiance is ($28.1W/m^2 < H < 329\ W/m^2$).

III. PHOTOVOLTAIC PANEL

The electrical analog model of an ST5 solar panel from Siemens was made, using Spice [1] within the development framework of Mentor Graphics, from which the circuit described in Fig. (1) was obtained. For its simulation the following equivalences were used: $1W/m^2$ which will be equivalent to 1V to irradiance, as well as $1^{\circ}C$ equal to 1V for temperature and 1 day will be equivalent to 1us thus the maximum time corresponds to an equivalent number of days, since a transient analysis was made to obtain the electrical parameters: short circuit current (I_{SC}) and the open circuit voltage (V_{OC}) Fig. (2b).

Table 1. Electrical characteristics of the module solar ST5

Maximum power P_{max} [W]	5
Rated current I_{MPP} [A]	0.32
Rated voltage V_{MPP} [V]	15.6
Short circuit current I_{SC} [A]	0.37
Open circuit voltage V_{OC} [V]	21.0

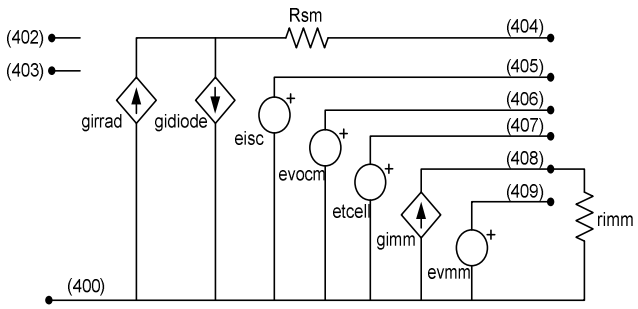

Fig 1. Electric analog model used to emulate the behavior of the solar module ST5, where the node (402) corresponds to irradiance, temperature (403) node, the node (405) to short circuit current and node (406) to the open circuit voltage.

IEEE Catalog Number CFP13827-ART
ISBN 978-1-4799-1461-6
978-1-4799-1461-6/13/$31.00 ©2013 IEEE

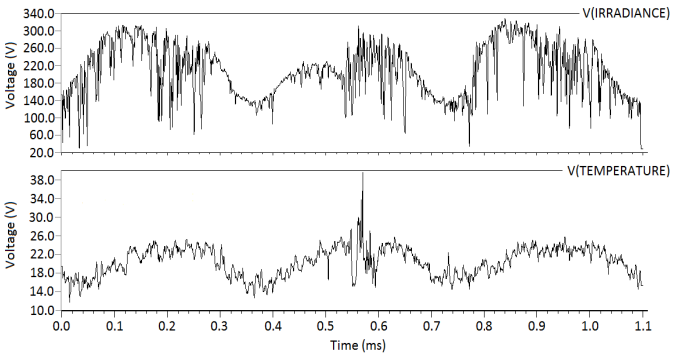

Fig 2a. Transitory analysis of the input nodes: (402), (403).

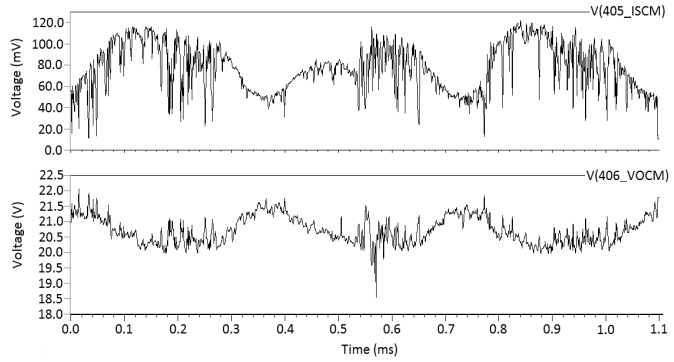

Fig 2b. Transitory analysis of the output nodes: (405) and (406).

IV. ARTFICIAL NEURAL NETWORK

Once the irradiance and temperature input values were obtained from the data base and the short circuit current and open circuit voltage were estimated from the analog electric model of the panel, we used the neural network toolbox of Matlab to create the multilayer artificial neural network as shown in Fig. (3), which is a feedforward network of (2-7-9-2) neurons having an input vector of two components, and an output vector of two components too. The network was trained to behave as a function aproximator using a supervised training and the Levenberg-Marquardt algorithm [2] to estimate the best weights and biases.

The training subset was obtained from the 70% of the original database and was used within the training algorithm to determine the weights and biases during the off line training. A second subset of 15% of the database was used as a test subset aimed to validate the training.

The neural network has a total of 18 processing units (neurons) where each processing unit implements the summation of the pondered inputs plus a bias as can be expressed by equation (1).

$$y_i = f\left[\left(\sum_i w_{ij}.x_{ij}\right) + b_j\right] \qquad (1)$$

The result of (1) is applied to a nonlinear tangent function known as Sigmoid function represented by equation (2).

$$a = \frac{e^n - e^{-n}}{e^n + e^{-n}} \qquad (2)$$

The domain of this function is the set of real numbers and its range is limited by the interval [-1, 1], also this function has the advantage that it can be differentiable.

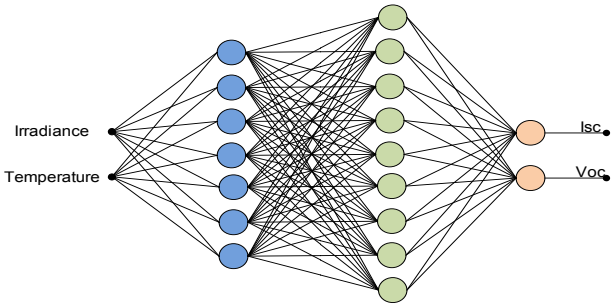

Fig 3. Artificial neural network implemented in Matlab.

In order to validate the training of the network, we used the graph of the mean squared error (MSE) Fig. (4) and the graph of the correlation coefficient (R) Fig. (5) from which the following table was obtained.

Table 2. Results of the training in Matlab

$MSE_{(Training)} = 1.2585E\text{-}4$	$R_{(Training)} = 0.99938$
$MSE_{(Validation)} = 1.4254E\text{-}4$	$R_{(Validation)} = 0.99953$
$MSE_{(Test)} = 1.7971E\text{-}4$	$R_{(Test)} = 0.99879$

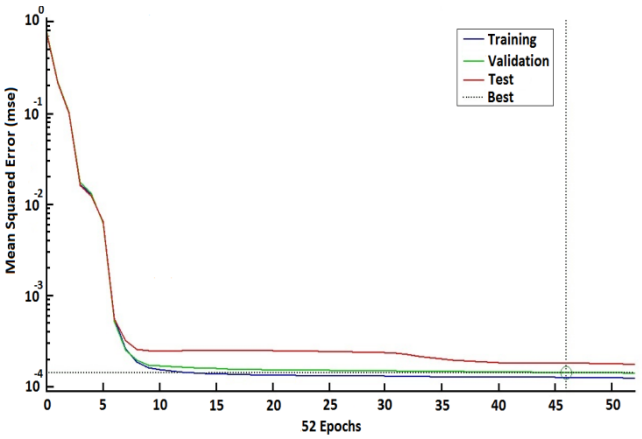

Fig 4. Graph of the average quadratic error (MSE) for the sets of training, validation and test.

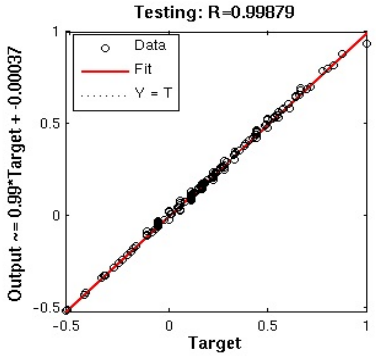

Fig 5. Graph of the correlation coefficient (R) for the set of test.

IEEE Catalog Number CFP13827-ART
ISBN 978-1-4799-1461-6

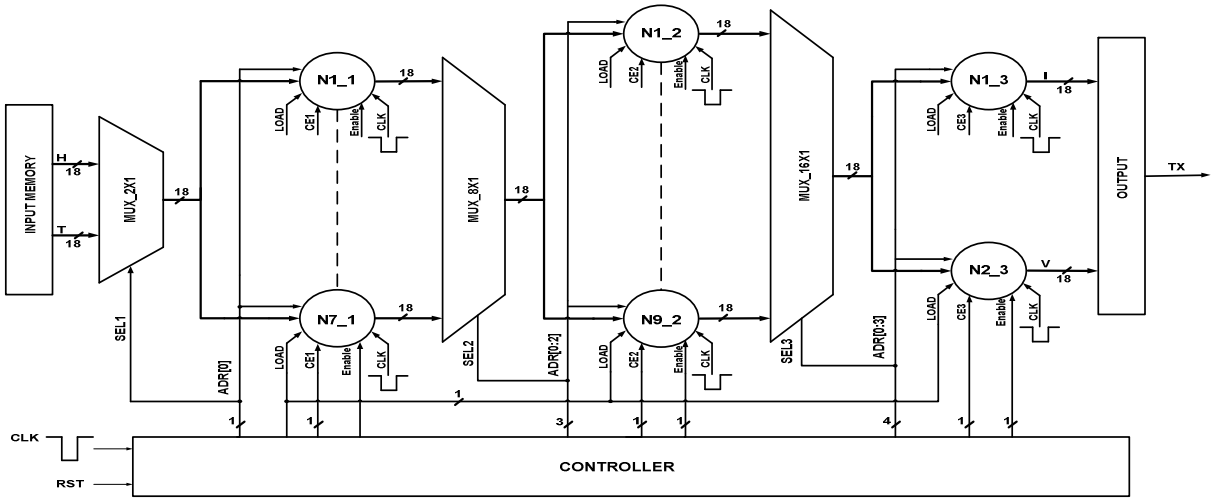

Fig 6. General diagram of the PV module implemented in the FPGA.

V. FPGA IMPLEMENTATION

Once the training of our artificial neural network in Matlab was validated to find the optimal values of weights and biases that would produce the minimum error between the output of the artificial neural network and the target values, then the implementation of the neural network was made using VHDL hardware description language [3-6], and ISE version 14.2 of XILINX on a digital FGPA XC6SL45 device of the Spartan - 6 family. Fig. (6) shows the network general block diagram of the digital implementation.

The hardware implementation of the ANN in the FPGA device is based on two main aspects: the simple neuron function and the controller.

As can be seen in Fig. (6) the general diagram of the PV module implemented in the FPGA is composed of several blocks such as:

- *Simple neuron function blocks:* that will allow us to make the processing of input values.
- *Multiplexers:* they will allow entries to different layers, so they can be processed by neurons.
- *Input memory block:* this block will be comprised of two read only RAM memories, where we have stored the data of irradiance (H) and temperature (T), it is worth mentioning that to test the response of the artificial neural network in the FPGA the test subset was used just like it was used in Matlab.
- *Output block:* this block will have two RAM memories configured in the read write mode where the output values of current and voltage resulting from the trained neural network can be stored to be transmitted using the serial transmission protocol at 9600 bauds to any computer to post process these results and compare them using Matlab with the response of the ST5 panel.
- *Controller:* This block will control the flow of information through the different layers that compose the neural network. Besides carrying the control of the blocks described above, through their signals, we will be able to increase the time response of the ANN, applying the pipeline technique.

FPGA Implementation of a simple neuron

As can be seen in Fig. (7) a neuron described in the FPGA will be comprised of three fundamental units: storage (ROM) unit, unit of calculation of the weighted sum (MAC), unit of approximation to the activation function (AF).

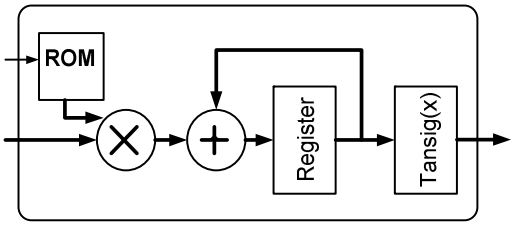

Fig 7. Blocks that makes up a single neuron.

Unit of storage (ROM)

In this ROM memory we will store the weights and the biases of each neuron. They will be represented by 18-bit words, since these values will be represented with fixed point, 9 bits for the integer part and 9 bits for the fractional part. The size of this ROM memory will be defined according to the position of the neuron in the network and the number of entries; this is represented by the equation (3).

IEEE Catalog Number CFP13827-ART
ISBN 978-1-4799-1461-6

$$T_{ROM(i)} = N_{neuron(i-1)} + 1 \quad (3)$$

Where:
i - Is the number of the layer.
$T_{ROM(i)}$ - Is the size of the layer ROM i.
$N_{neuron(i-1)}$ - Is the number of neurons in the layer $(i-1)$.

Calculation unit of the weighted sum (MAC)

This unit was implemented using an 18-bit multiplier that was associated with an adder, so that the sum can be estimated in this manner.

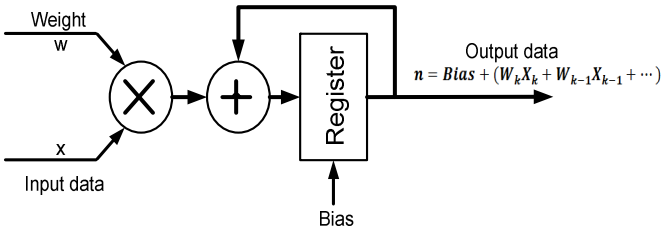

Fig 8. Multiplier accumulator module (MAC).

Activation function approximation unit

To represent the sigmoid tangent function in the FPGA a linear approximation was performed from a combination of segments of straight lines of the form $y = ax + b$, thus several intervals were used $[C_i, C_{i+1}]$, the slope (a) and the ordinate at the origin (b) were evaluated for each line segment, so we have a function evaluated by intervals to estimate (y).

$$f(x_i) = a_i x_i + b_i \quad \text{si} \quad x_i \in [C_i, C_{i+1}]$$

$$f(x_i) = 1 \qquad \text{if } x_i > 3 \quad (4)$$

A total of 16 lines were used to represent the tangent sigmoid function. Fig. (9) shows the results of the activation function under the *ModelSim* simulation platform. Fig. (10) shows the comparison between the Matlab produced activation function and the VHDL produced function with the line segments. Fig. (11) shows the error percentage obtained between these two functions, thus obtaining an maximum error of 0.8196%.

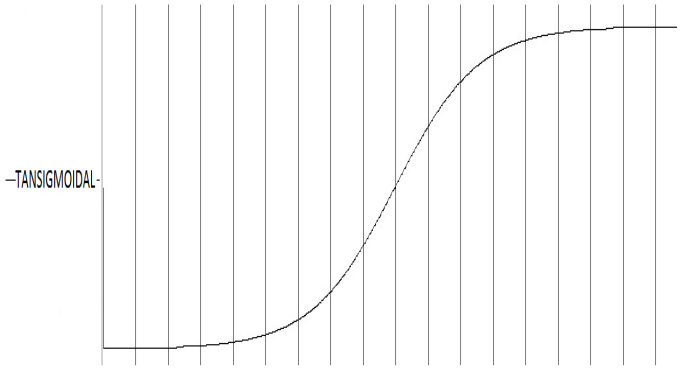

Fig 9. Activation function implemented in VHDL

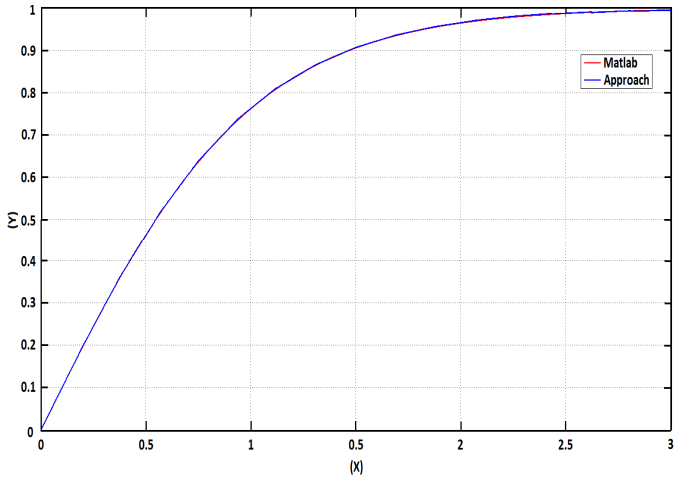

Fig 10. Comparison of the positive part of the activation function calculated in Matlab and simulated in VHDL.

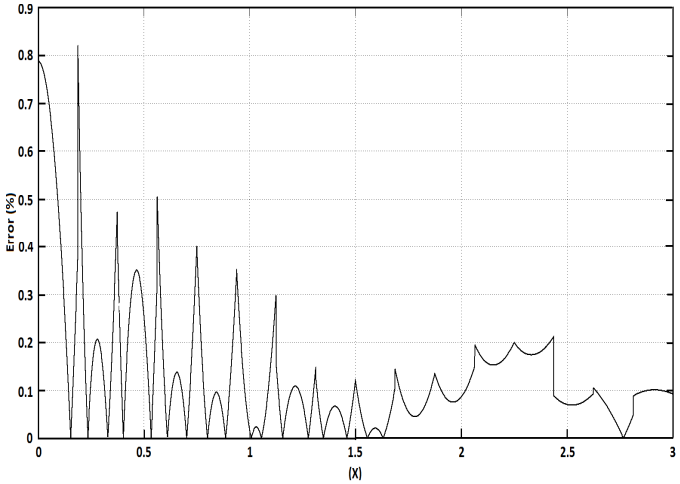

Fig 11. Graph of the error between the activation function estimated in Matlab and simulated in VHDL.

Pipeline Technique

Using the controller signals: CE1, CE2 and CE3 it is possible to make the 3 layers of the network to work at the same time, making the first data vector (irradiance and temperature values) to be read from the input memory at the first processing cycle, and then processed by the 7 neurons in the first layer; at this same first cycle neurons in layer 2 and 3 are processing erroneous information. In the second processing cycle the first layer output is processed by the second layer and layer one is processing the second data vector (irradiance and temperature values) form the input memory so on and so forth, until the fourth processing cycle in which the correct values of voltage and current are produced. The processing time response of the neural network without using this pipeline technique is 2100ns and it takes around 476000 operations per second, in contrast, with the pipeline technique the response obtained is 1000ns and one million of operations per second.

IEEE Catalog Number CFP13827-ART
ISBN 978-1-4799-1461-6

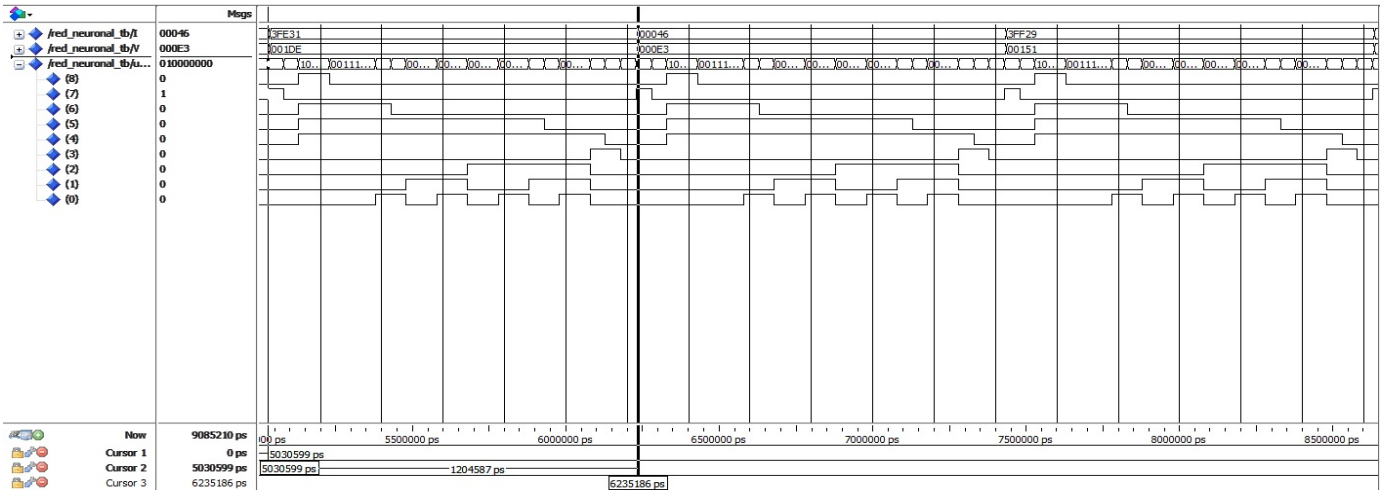

Fig 12. Simulation results from the ANN implemented with FPGA.

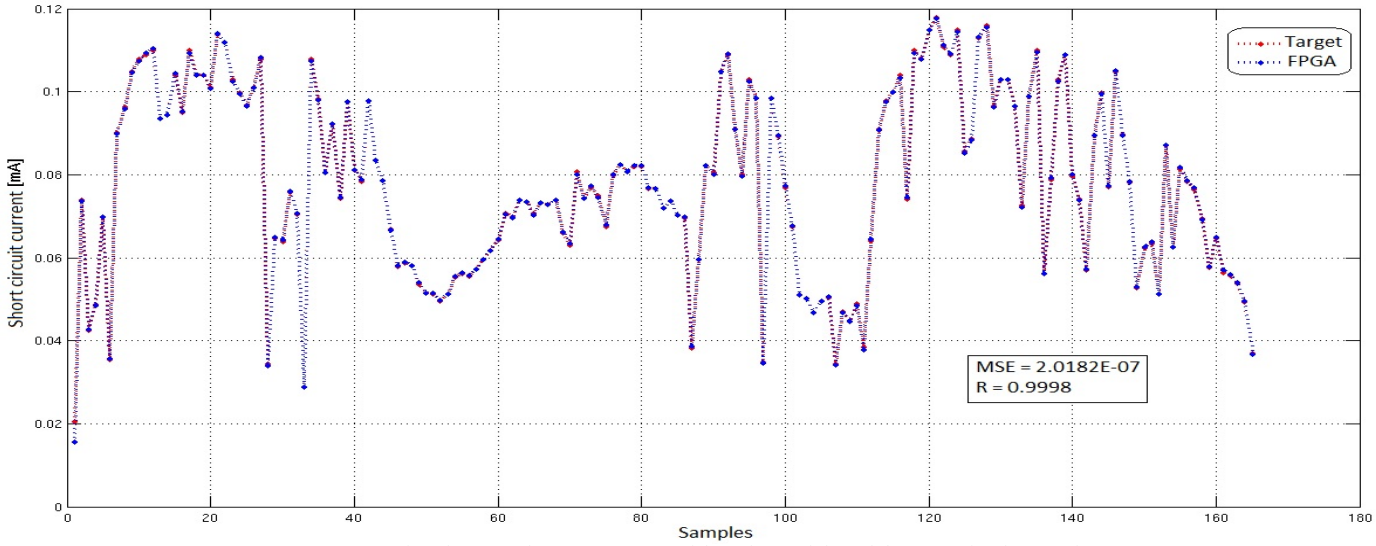

Fig 13. Comparison between the output current of the solar module and the FPGA implementation.

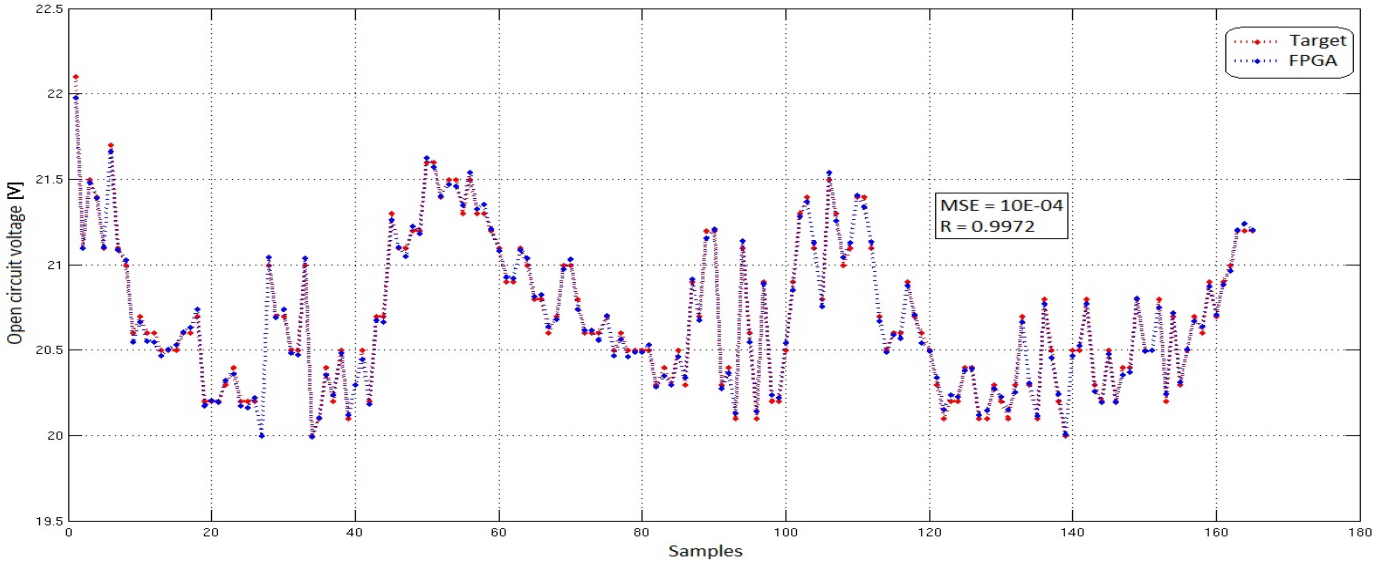

Fig 14. Comparison between the voltage response of the solar module and the FPGA implementation

IEEE Catalog Number CFP13827-ART
ISBN 978-1-4799-1461-6

The MSE of the Fig.(13) and Fig. (14) was calculated using the equation (5).

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{Y}_i - Y_i\right)^2 \quad (5)$$

Where:
N - Is the total number of samples.
$\hat{Y}_i$ - n-th output value of the model solar.
Y_i - n-th output value of the FPGA.

And the correlation coefficient of Pearson R was calculated using the equation (6).

$$R = \frac{N\sum_{i=1}^{N} X_i Y_i - \sum_{i=1}^{N} X_i \sum_{i=1}^{N} Y_i}{\sqrt{N\sum_{i=1}^{N} X_i^2 - (\sum_{i=1}^{N} X_i)^2}\sqrt{N\sum_{i=1}^{N} Y_i^2 - (\sum_{i=1}^{N} Y_i)^2}} \quad (6)$$

Where:
N - Is the total number of samples.
X_i - n-th output value of the model solar.
Y_i - n-th output value of the FPGA.

Table 3. Resource used for the FPGA

Slice Logic Utilization	Used	Available	Utilization
Number of Slice Registers	977	54576	1%
Number of Slice LUTs	2857	27288	10%
Number of occupied Slices	1029	6822	15%
Number of bonded IOBs	41	218	18%
Number of RAM B16BWERs	2	116	1%
Number of RAM B8BWERs	2	232	1%
Number of BUFIO2/BUFIO2_2CLKs	1	32	3%
Number of OLOGIC2/OSERDES2s	36	376	9%
Number of DSP48A1s	36	58	62%
Number of PLL_ADVs	1	4	25%

VI. RESULTS

As can be observed, considering an electrical analog model defined in Spice as that shown in Fig. (1), it was possible to emulate the behavior of an ST5 solar panel from Siemens, which represents 33 serially connected cells from which the electrical parameters could be extracted (short circuit current and open circuit voltage). The architecture of the proposed ANN is shown in Fig. (3) from which the best results were found, from an evaluation of the error between the input to the ANN and the expected value of the solar cell panel. Also it can be observed that the ANN architecture is a good function aproximator since it could effectively abstract any proposed profile. Also the Levenberg Marquardt algorithm is suitable for this sort of architectures, in which a minimum number of neurons is required to have a faster processing, obtaining a training time of around 3 seconds. As can be seen from Fig.(13) and Fig. (14) there is a good approximation between the measured results of the solar panel and those expected from the ANN digital implementation, from which an error of MSE = 2.0182E-07 and a factor of correlation R = 0.9998 for the current were obtained and a MSE = 10E-04 and a R=0.9972 for the voltage were obtained.
In the Table (3) we can see the small amount of resources used for the device FPGA, where the number of DSP's was considerable, because the implementation used 18 DSP's to perform the product between the input and the weight and other 18DSP's were used to evaluate the equation (4) in each neuron.

VII. CONCLUSIONS

It was possible to implement a solar panel in a digital device FPGA, only needing two environmental parameters (irradiance, temperature) having in this way a low computational effort, besides this emulator can operate in real time being a, reason why this type of devices are very useful to make the analysis of a solar module in a research laboratory, in addition that the FPGA used for this application is very viable due to the low consumption of resources which were used and to its type of architecture which allowed the parallelism with which the biological neural networks work, giving therefore the flexibility to make the implementation of a whole photovoltaic system in the future.

ACKNOWLEDGEMENTS

This work was supported by CONACYT also want to thank Dr. Oliverio Arellano Cárdenas, M. en C. Luis Martín Flores Nava and M. en C. Omar Hernández Garnica for their assistance in programming.

REFERENCES

[1] L. Castañer, S. Silvestre "Modelling photovoltaic systems using PSpice". University Politecnica of Cataluña, Barcelona, Spain 2002; pp. 77 - 94.

[2] M.T. Hagan, H. B. Demuth, M. Beale "Neural Network Design" Oklahoma State University pp. 11-1 – 12-28

[3] A. Mellit, H. Mekki, A. Messai, H. Salhi "FPGA - based implementation of an intelligent simulator for stand- alone photovoltaic system". Expert Systems with Applications 2010; 37: 6036-6051.

[4] H. Mekki, A. Mellit , S. A. Kalogirou, A. Messai, G. Furlan "FPGA - based implementation of a real time photovoltaic module simulator". Progress in photovoltaics: Research and Applications 2010; 18: 115 - 127.

[5] H. Mekki, A. Mellit, H. Salhi, B. Khaled "FPGA - based implementation of multilayer perceptron for modelling of photovoltaic panel". In American Institute of Physics (AIP) Conference Proceedings 2008; 1019: 211 - 215

.

[6] E. Koutroulis K. Kalaitzakis, V. Tzitzilonis "Development of an FPGA-based system for real-time simulation of photovoltaic modules". Microelectronics Journal 2009; 40: 1094 - 1102.

IEEE Catalog Number CFP13827-ART
ISBN 978-1-4799-1461-6
978-1-4799-1461-6/13/$31.00 ©2013 IEEE