May-June 2020
ISSN: 0193-4120 Page No. 3474-3476

ASIC Implementation of Random Perturbation Algorithm for Neural Network Application

Dharamvir[1], Arul Kumar V[2]

[1]Research Scholar, [2]Asst. Professor
[1,2]School of CSA, REVA University, Bengaluru, India

Article Info
Volume 83
Page Number: 3474-3476
Publication Issue:
May-June 2020

Article History
Article Received: *19 August2019*
Revised: *27 November 2019*
Accepted: *29 January 2019*
Publication: *12 May 2020*

Abstract
An Analog VLSI Implementation of an on-chip learning neural network is described in this paper. The network considered comprises an analog feed forward network with digital weights and update circuitry. The chip consists of a comparator, incrementer and decrementer circuit to update the weights. The training algorithm used is Random Perturbation Algorithm. From experimental results it is seen that the weights are updated as per the algorithm. Intense simulations were carried out in HSPICE simulation tool using 0.18µm technology to verify its functioning.

Keywords : *Neural Networks, Perturbation Algorithms, Incrementer, Data Connectivity, Image Transformations*

1. Introduction

One of the major impediments to implement an artificial neural network is the complexity of the hardware required to implement the training algorithm.[1,2]. A VLSI neural network can be applied in many situations requiring fast, low power operations [3].

Artificial Neural Networks require to be trained to solve any problem. The training can be either offline or online. The offline training involves using the weights generated by simulation. The weights are represented using fixed number of bits for hardware implementation. This results in quantization error. On chip learning can adjust the weights to obtain the desired functionality and reduce the quantization error.

The back propagation method as well as Random Perturbation has been used in hardware implementation for training the network. The back propagation method involves implementation of circuits to estimate mean square error and derivatives of the error function[4]. This results in a complex circuit. A simple method of implementing training by Random Perturbation has been reported by Kush etal. The neural network is implemented as mixed signal circuits using multiplying DAC. The analog circuit is used to implement the weight perturbation method. Here we report an alternative method to perturb the weight using partly digital implementation based on incrementer and decrementer circuits. The weights are stored as charge on capacitors. This requires frequent refreshing as capacitors loose their charge over time. The weights are updated bit by bit and are stored in the memory. Thus the network needs to be trained only once or only when a new number needs to be solved. The second section gives the basic algorithm of Random Perturbation. Third section looks into the hardware implementation of the training algorithm. The fourth section analysis the test results obtained and the finally the conclusion is provided in the 5th section.

2. Random Perturbation Algorithm

The artificial neural network consists of a processor, control circuit, a training algorithm and a memory to store the weights as shown in figure 1.

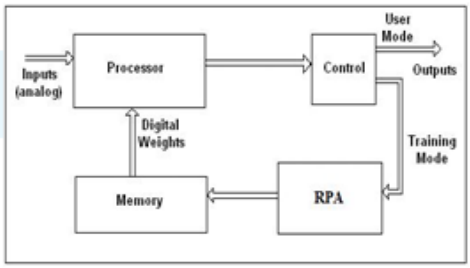

Figure 1: Neural Network Block Diagram

May-June 2020
ISSN: 0193-4120 Page No. 3474-3476

The processor consists of an analog multiplier and a summer. There are different architectures of analog neural network reported earlier in literature [5,6,7]. The control circuit decides whether neural network will be operated in user mode or training mode. Once training mode is selected the training algorithm is executed and finally the updated weights are stored in memory.

The neural network is trained here by using the Random Perturbation Algorithm also called as parallel weight update rule[8]. The algorithm begins with the generation of random weights and then adjusts the weights during every iteration. The following is an outline of the algorithm [1]

Initialize Weights;
Get the error; While (error > error goal);
Perturb Weights;
Get New Error;
If (New Error < Error), Weights=New Weights; Error = New Error; Else Restore Old Weights; End End
The flow of the algorithm has been implemented using the flow chart shown in figure 2.

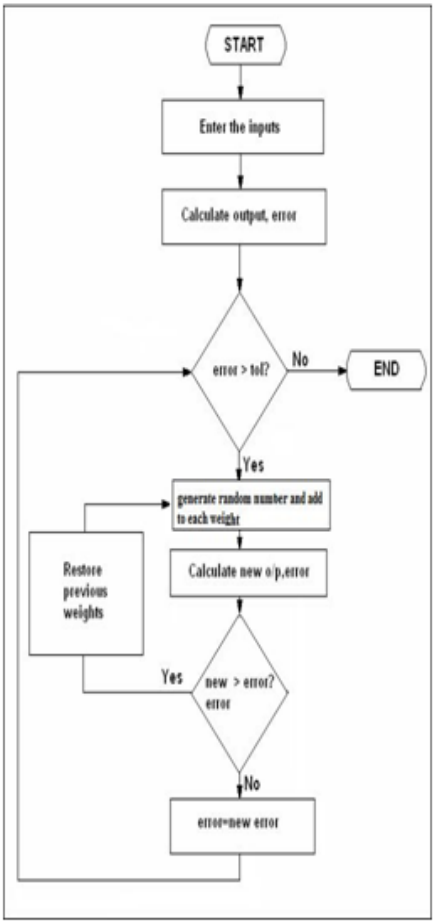

Figure 2: Flowchart of RPA

Hardware implementation

The hardware implementation of the algorithm includes an operational amplifier, and an incrementer and decrementer circuits as shown in figure 3.

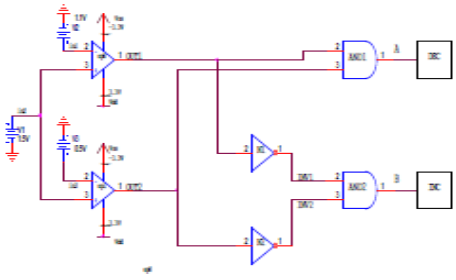

Figure 3: Hardware implementation of RPA

Op Amp as a Comparator

The circuit diagram of the two stage Op Amp is shown in figure 4.

The two stage Op Amp is used as a comparator. The first Op Amp is used to compare the input voltage V1 with the target plus tolerance voltage V2. The output OUT1 of this Op Amp is high if the V1 is greater than the V2. Else it is low.

The second Op Amp is used to compare the input Voltage V1 with the target minus tolerance Voltage V3. The output OUT2 of this Op Amp is high if the V1 is greater than the V3. Else it is low. The operational amplifier is designed to meet low power dissipation requirements of analog circuits [9].

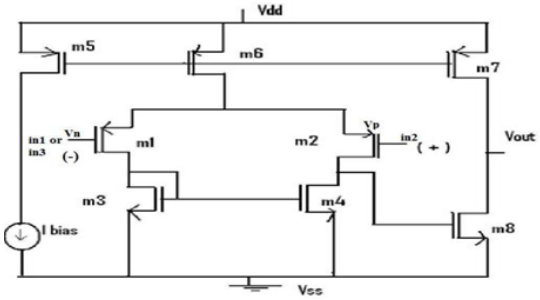

Figure 4:Two Stage Op Amp

Decrementer Circuit (DEC)

The gate level description of a one bit incrementer is shown in figure 5.

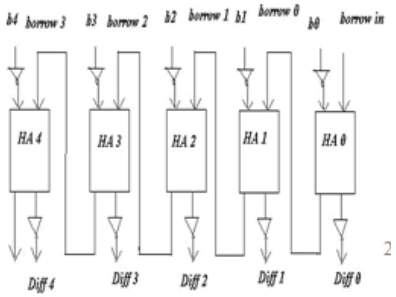

Figure 5: One bit Decrementer Circuit

The decrementer circuit has been implemented to carry out two's complement subtraction. The circuit decrements the LSB bit of the weight one bit every iteration if the input is higher than the target plus

May-June 2020
ISSN: 0193-4120 Page No. 3474-3476

tolerance voltage. If the input voltage V1 is greater than target plus tolerance voltage V2 then it V1 is also greater than target minus tolerance VoltageV3. Thus if OUT1 and OUT2 are both high then the decrementer is executed and the weight is decremented by one bit.

Incrementer Circuit (INC)

The gate level description of a one bit Incrementer is shown in figure 6.

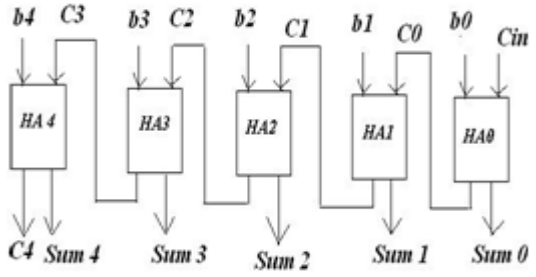

Figure 6: One bit Incrementer Circuit

The circuit increments the LSB bit of the weight one bit every iteration if the input is lesser than the target plus tolerance voltage. It is obvious that if the input voltage V1 is lesser than target minus tolerance voltage V3 then V1 is also lesser than target plus tolerance VoltageV2. Thus if OUT1 and Out2 are both low then incrementer is executed and the weight is incremented by one bit.

The incrementer and decrementer circuit consist of a series of Half Adders (HA) arranged in a carry ripple fashion. In this work the Random Perturbation Algorithm is implemented to update 5 bit weight [5,6,7 Hence five Half Adders are used in the incrementer and decrementer circuits.

3. Test Results

Intense simulations were carried out in HSPICE simulation tool using 0.18µm technology to verify the functioning of the hardware mentioned in section 3 of this paper. For various combinations of the input voltage, the various outputs have been recorded as shown in table 1.

Table 1: Output details at each stage of RPA

V1	V2	V3	OUT1	OUT2	A	B
1.5	1.1	0.5	3.3	3.3	3.3	0
1.1	1.1	0.5	-0.23	3.3	0	0
0.5	1.1	0.5	-3.3	-0.55	0	0
0.4	1.1	0.5	-3.3	-3.3	0	3.3

Hardware Implementation

As explained in section 3, the weights have to be updated if the new error is greater than old error. Here the weight is decremented if the above mentioned case is true as is seen in table 1 where when V1 is greater than V2, V1 is also greater than V3 and so OUT1 is and OUT2 are high indicated by 3.3V (logic 1) and hence A is high and B is low triggering the decrementer circuit to decrement the weight. Also the weights are updated if the input voltage V1 is lesser then target minus tolerance Voltage V3 and hence V1 is also lesser than V2, in which case the weight is incremented by one as is obvious from the table1, when V1 0.4V and V2 is 1.1V and V3 is 0.5V, OUT1 and OUT2 both are low (-3.3V Logic 0) and B is high and A is low. This triggers the incrementer circuit and the weight is incremented by one bit. If the input voltage is anywhere in between the target minus tolerance Voltage and target plus tolerance Voltage, the outputs A and B both are zero and neither the incrementer nor the decrementer are executed. Thus the old weights are retained as per the Random Perturbation Algorit.

4. Conclusion

A VLSI implementation of an on Chip training algorithm for artificial neural network is demonstrated in this paper. The Random Perturbation Algorithm is implemented using Op Amp as a comparator and the decrementer and incrementer to decrease and increase the weights respectively. The simulation results show that the training circuit follows the algorithm to a large extent. The can be extended to update the weights constantly and on the fly.

References

[1] Vincent F. Koosh, Rodney Goodman, "VLSI Neural Network with Digital Weights and Analog Multipliers", 0-7803-6685-9/01/ IEEE, 2018

[2] Paul W. Hollis, John J.Paulos," A Neural Network Learning Algorithm Taiolored for VLSI Implementation", IEEE Transactions on Neural Networks, Vol 5, September 2014

[3] R.Coggins, M.Jabri, B.Flower and S.Pickard, "A Hybrid Analog and digital VLSI Neural Network for Intracardiac Morphology Classification", IEEE J. of Solid –Sate Circuits, vol.30, no.5, pp.542-550, May 2017.

[4] Perry D Moerland, Emile Fiester, " Neural Network adaptations to hardware implementations", in Handbook of Neural System Data Classification with Computation, 10P Publishing and oxford university press, 2018, 97/1, pp E1.2:1

[5] Sujith Sudarshan, Santosh K and S.L.Pinjare, "MDAC Synapse – Neuron for Analog Neural Networks", National Conference on convergence ofsignal processing, communication and VLSI Design, NCSCV10-13, 186, 2019.