DOLFIN - Digit Online For Integration Neural Networks

Andreas Wassatsch, Marc Haase, Dirk Timmermann

University of Rostock, Dep. EE and IT, Institute of Applied Microelectronics and CS
R.-Wagner-Str. 31, D-18119 Rostock, Germany, wassatsch@e-technik.uni-rostock.de

ABSTRACT

In this paper we describe an approach for using digit online arithmetic on the field of neural network computation. Digit online, a serial most significant digit first arithmetic, shows significant advantages over all other digital implementations. The serial communication between the online modules make the implementation of connection intensive networks feasible. The accuracy of the computation is only loosely coupled with the chosen digit level range, which determine the necessary count of interconnections. Furthermore the accuracy is eligible through the length of the processed digit vector. The goal of this paper is to develop a strategy for the implementation of different network models. The comparison with the results of other implementations illustrate the advantages of the digit online approaches and the suitability for the application on the field of neural networks.

1. INTRODUCTION

Artificial neural networks are an excellent example for massively parallel data processing. The implementation of fully parallel data processing is mostly limited by resources available. For this reason commonly only a small inner loop of the whole data processing algorithm is implemented which was been previously identified by analysis of the resources required during the calculation. To fulfill the algorithm the operations will be sequential processed with assistance by the implemented shared resource. This leads to a reduced performance of the resulting system.

The presented approach shows a possible solution for this problem on the area of neural networks. After a short discussion of previous approaches and an introduction to the basics of neural networks and digit online arithmetic we will give a survey of neural networks implementation using digit online arithmetic in section 2. In section 3 we will demonstrated the usability of our approach in some examples. At last, we will discuss our results in section 4.

1.1. Previous approaches

An overview of different approaches for implementing neural networks can be found at [1]. Like neural networks themselves the different implementation approaches can be classified by numerous characteristic properties. On the implementation level the representation and processing of values differs for the various approaches.

1.1.1. Analog realizations

One of the best known analog integrated neural network chips is the Intel 80170NX [2] which uses EEPROM-cells to address the main problem of analog implementations, the storage of weight factors. The chip integrates 64 analog neurons and 10260 trainable weights and achieves a performance of 1.3 giga connections per second (GCPS). The excellent result of 80 GCPS by 256 neurons and 8192 weights of the AT&T chip [3] is accomplished by sacrificing some accuracy in the input and weight vectors. Furthermore, on-chip learning functionality is not implemented in this device.

1.1.2. Digital Implementations

Due to the wide area of digital implementations of neural networks only some representative examples will be mentioned. The majority of them implement the calculation intensive part of the computing algorithm only. Therefore, the results of the analog and digital implementations are difficult to compare. In addition, information on the number of multiplications and additions per second is seldom available. Examples of digital implementations are the systolic array approach like the Synapse MA16 [4] and the SAND/1-architecture [5]. Both architectures are used to speedup the matrix operations like weight-multiplication and summation of the activations for a neural network. Specialized on the RBF-like (Radial Basis Function) the ZISC [6] represents an user programmable device for the realization of neural network applications. In [7] the borrow-save number system has been chosen which is one reason for the comparatively low performance of this digit serial approach. The choice of this system is determined by the different target architecture. An other reason is the limited architecture to compute only the functions for one neuron, which not consequently use the potential of the online algorithms. There also exist hybrid implementations which combine the advantages of both implementation styles.

1.2. Neural Networks

Due to the numerous literature available on principles and implementations of neural networks we outline only those topics which are relevant for our solution.

The operation on a neural network can be divided in general into two phases, the training or learning task and the evaluation task. A learning phase is necessary because the fixed storage of weights for evaluation of the synaptic connections is unusual and severely limits the adaptation of the network to changing conditions. Commonly, external memory served for storing some sort of learning patterns. Thus a different application requires a renewed content of the pattern memory only. In the original implementation of the neuron the activation function decides whether or not the neuron is activated. The leads to a leap function for the output and maps the possible output to two discrete values. In a multilayer network the weight multiplier of the following neurons can be simplified for such an output signal. Unfortunately, this disturbs the calculation flow in the network and reduces also the information about the strange of the activation. Therefore, mostly non-discrete activation functions are used.

1.3. Digit Online - MSB first serial arithmetic

Unlike most other serial arithmetic implementations digit online algorithms generate results starting with the most significant digit first. This is possible due to the redundant number representation employed. For our implementation we prefer a signed digit (SD) representation but carry save (CS) is also usable. Simple arithmetic functions like addition or multiplication can be implemented without redundant adders in least significant digit first manner. A MSD-first implementation is also possible when using redundant adders. Higher level arithmetic operations like division or square root are MSD-first algorithms by nature and can not be transformed to LSD-first without an extreme performance penalty. Due to the result generation starting with the most significant digit we can easily integrate this algorithm. The idea of digit online algorithms, first presented in [8], can be described for a two operand example with equ. 1 by equ. 2 and the following iteration equ. 3

$$S = f(X, Y) \text{ with } X = \sum_{i=0}^{j+\delta} x_i r^{-i};\ Y = \sum_{i=0}^{j+\delta} y_i r^{-i} \tag{1}$$

$$\left\{x_i, y_i, s_i\right\} \in \{-p, \ldots, -1, 0, 1, \ldots p\}$$

$$\text{and } \frac{r}{2} \leq p \leq r-1$$

Initialization:

$$\begin{array}{rlrl} W_{(-\delta+1 \ldots -1)} & \leftarrow C & X_0 & \leftarrow 0 \\ s_{(-\delta \ldots -1)} & \leftarrow 0 & Y_0 & \leftarrow 0 \end{array} \tag{2}$$

Recursion:

$$\begin{array}{rcl} \text{for} & & j = 0, 1, \ldots, m \\ W_j & \leftarrow & (W_{j-1} - s_{j-1}) + \\ & & f(x^{-\delta}, x_j, X(j-1), y_j, Y_{j-1}) \\ s_{j-1} & \leftarrow & S(W_j) \end{array} \tag{3}$$

and the selection function:

$$S(W_j) = \begin{cases} r-1, & if \quad w_s < W_j \\ 0, & if \quad -w_s \leq W_j \leq w_s \\ -r-1, & if \quad W_j < w_s \end{cases} \tag{4}$$

The algorithm starts with an initialization of some variables, the scaled residual W_j, the operand vectors X_j and Y_j, and the first δ-result digits s. Depending on the function the algorithm needs δ leading operand-digits to calculate the first result digit. This incorporates a function specific online delay δ for the result generation. During the following recursion the actual residual W_j will be calculated and each iteration outputs a new result digit s_j determined by the selection function equ. 4. The resulting digit vector approximates the result step-by-step. The redundant number system allows any necessary corrections of a previously estimated result.

By cascading digit online modules as shown in Fig. 1 sequential operations can be overlapped [9]. Compared to conventional implementations of successively LSD-first and MSD-first operations this results in drastic performance improvements.

2. DIGIT ONLINE IMPLEMENTATION OF NEURAL NETWORKS

We now introduce our approach for the VLSI-implementation of neural networks. Usually, the evaluation task of a neural network

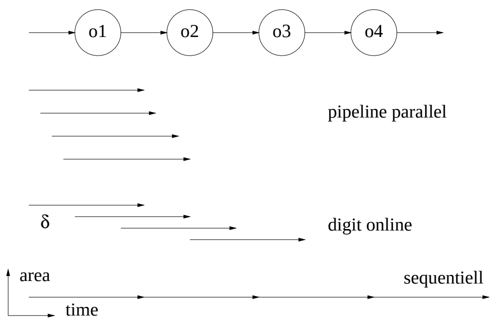

Figure 1: Cascading of digit online modules

does not require an MSD-first arithmetic implementation. Only some specialized activation functions and the learning task recommend themselves to applying digit online algorithms as they require naturally MSD-first functions like division or square root. In favor of a small library of basic processing modules we use digit online in the evaluation phase as well. With respect to resources two different implementations are possible. The first one is a full performance architecture with dedicated data paths for evaluation and learning at the same time. Unfortunately, we can not reuse the weight memories and need additional registers. One possible solution to fully parallelize evaluation and learning is to accumulate the weight changes over the whole learning pattern set and computing the arithmetic mean of the weight update. By utilization of this algorithm we must consider the influence on behavior of the learning process. The second architecture reuses the components for both phases which reduces the performance of the system but at the same time the circuit effort as well. Switching between two phases we also introduce an alternation of the data-flow direction in our design. Therefore, we must implement additional structures to adapt the connection ports of the modules to the changing conditions which means a switch between block inputs and outputs.

2.1. Concept

The implementation of our approach is according to the concept shown in Fig. 2. Classification of the input vectors will be done by a slightly modified digit online multiplier which calculates the product of a parallel weight vector and the serial stimuli vector of the synaptic connection of the preceding neuron. This simplification reduces the area by about one half at the same digit online delay δ of two compared to a normal digit online multiplier. The accumulation of the sum on weighted stimulation of each neuron is done in the following using a balanced digit online adder. The online delay increases with the number of inputs. Balancing is necessary to prevent data synchronization problems. The neural network developer should not be responsible for the varying delay of signals depending on the number of stimulation network inputs as imposed by the specific type of arithmetic used. At least through the different activation functions the stimulation of the neuron can be classified. The above mentioned data processing path describes the calculations in one neuron during the evaluation phase. For the learning task we needs a second data-path which can like mentioned above be implemented by reuse of the evaluation path components or be realized separate from them to achieve a higher throughput on the system. Using this basic structure of a neuron we can implement different network types with marginal changes for specialized architectures like direct feedback or fully connected networks. The whole neural network implemented with digit online is surrounded by parallel-serial converters for all inputs and on-the-fly conversion modules for the serial-parallel transformation at all outputs of the network. Applying this shell of conversion elements our approach is adequate to a conventional parallel approach from a system point of view and can be as easily used as parallel architectures at much less interconnection requirements. The additional components like the test and learning pattern memory and the control unit for the neural network working phases are arranged outside this block.

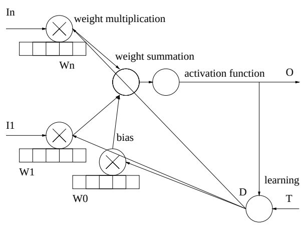

Figure 2: Concept of digit online nn

2.2. Activation function

The implementation of the activation function is one of the most critical parts during the design of neural networks. In the majority of the previously presented digital implementations this function is not implemented as they are mainly used as an accelerator for the huge number of multiply and add operations. In those cases, the determination of the activation of each neuron will be done at a higher control level, mostly a microprocessor. When implemented in digital as in [7] it exhibits to be very expensive and feasible by table based approaches only. Commonly, functions as shown in Fig. 3 are used for activation. At first, it is the sigmoid function.The result of this function must be rescaled and transformed to fully exploit the possible number range of our modules like the tanh function. With equ. 5 we can approximate the behavior of both previously mentioned function with the advantage of a unified implementation.

$$f(x) = \begin{cases} 1 & if \quad g(x) > 1 \\ g(x) & if \quad -1 <= g(x) <= 1 \\ -1 & if \quad -1 > g(x) \end{cases} \tag{5}$$

$$\text{with} \quad g(x) = n * x \quad \text{or} \quad g(x) = x^{\frac{1}{3}}$$

In our approximation we can implement the selection function using a saturation algorithm. The specific advantage of our MSD first system is that it permits the implementation of such a behavior without any performance penalty. Due to the MSD-first processing of the digits we can compare the argument vector like in the real world which improves overall efficiency. For an LSD system we must wait for the whole digit vector to be processed before we can decide whether the neuron is activated or not. To simplify the integration with the other digit online modules the behavior of the saturation function follows the online concept. The application of our approximated activation function guarantees a digit online delay δ of 3 clock cycles for this submodule at a reduced implementation effort. The influence on the network characteristics in particular on the learning process due to the modified activation function can be compensated by additional learning cycles as shown in Fig. 4.

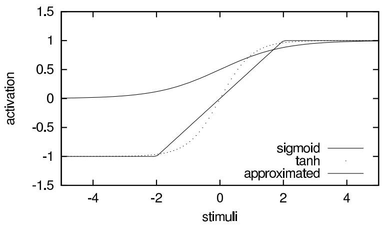

Figure 3: Plot of possible activation functions

The decision to increase the number of learning cycles at the benefit of reduced implementation effort depends on the application area of the neural network. If learning time is of less importance we can employ our smaller activation function.

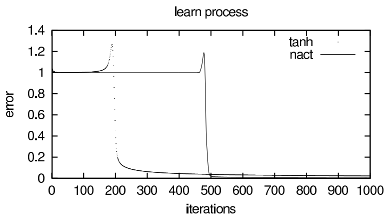

Figure 4: Influence of modified activation function on network behavior

2.3. Synchronization

The synchronization of the different modules becomes of significant importance during the design process especially in a system with serial data processing like our approach. The majority of digit online modules needs an information about the start and end of each operand if we cascade different modules. Specific actions like initialization of registers has to be done at this time. We need an adequate general solution to this problem to prevent re-implementation of synchronization during each design thus saving development time. A single synchronization bit per digit vector is sufficient but two or more sync digits are possible as well. Our mechanism is implemented as a shell around the whole digit online module to separate the digit online algorithm from the dataflow control. In this way we can reuse one of the possible zero digits representations for our synchronization intention. We can easily adapt the coding of the four possible character digits on the communication lines between the digit online modules as we use a high level description of our design based on VHDL. During the development process our synchronization scheme is helpful in identifying unbalanced data streams which lead to faults in data processing. Unfortunately, the proposed embedded synchronization in the data line costs at least one clock cycle reducing the throughput of the system. An alternative implementation of our synchronization mechanism with the advantage of a possible increased throughput is the utilization of an additional line for signaling synchronizations events between the modules. This solution needs more wiring and additional circuits. Furthermore, for both solutions of local synchronizations we pay for the simplicity of concatenation of the modules with the increased implementation effort which must be considered under the aspect of the number of parallel units like the neurons in one layer in our application area.

2.4. Performance analysis

As has been shown in Fig. 5 the throughput of the network is mainly determined by the highest digit online delay δ. The throughput is influenced by the observed growing of the vector length by some operations like the digit online addition. To reduce this effect the implemented activation function of each neuron should carry out also a limitation of the digit vector length.

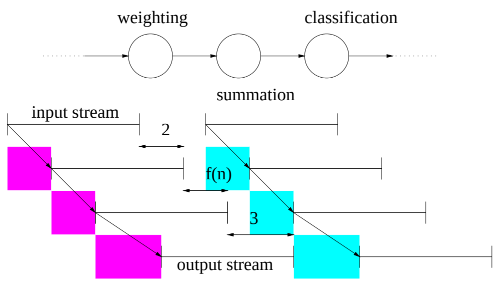

Figure 5: Analysis of the data-flow in the evaluation phase

2.5. Implementation results

For evaluating our digit online implementations we utilize a XILINX Virtex synthesis library. At first we verified the synthesis results of the basic digit online building blocks on the Virtex architecture. Furthermore synthesizing single neuron architectures with different parameters for input vector count and calculation accuracy produces the following results (Tab. 1). Based on this results we estimated the realisation effort and selected therefore the implementable architectures for the neural network.

input vec.	bit width	clb slices	func . gen.	flip flops	io pins	gate equ.
2	8	702	1403	671	71	14752
2	16	843	1683	735	135	16944
10	8	2223	4431	2169	215	46416
10	16	2646	5271	2361	407	52992
50	8	9260	18467	9043	935	192512
50	16	11082	22107	9875	1767	221008

Table 1: Synthese results for single neuron

3. NETWORK EXAMPLES

As example for the evaluation of our approach we realize different architectures of neural networks. Special attentation was dedicated to the applicability of the serial processing to become an optimized network performance on the evaluation phase. At first we try to solve the well known XOR-problem. This is an none trivial theoretical problem which can only be solved by higher order networks then single adenaline or preceptrons. A network with at least one hidden layer is required to fulfill this task. Therefore this problem is representative for the majority of neural networks architectures with the advantage of its simplicity. Because of the additional synchronization effort the possible implementation based on only four neurons are not be considered. In the first step we focused our investigation on implementation of the evaluation phase only. Therefore we realized a feed-forward network build up with fife neurons, two for input layer, two for the single hidden layer and one for result generation in the output layer. On this network architecture we investigated the applicability of our proposed activation function and compared the behavior of the networks. The different weight sets of the feed-forward network are calculated by an external application. After the successful evaluation of the feed-forward network in the evaluation phase we decided to implement the back-propagation rule. For the implementation we utilized the slower version characterized by the shared data-paths for evaluation and learning. Contrary to the preceding more academic examples which point out the basic applicability of our procedure, at least the example of a waveform detection network is more corresponding to real world application.

4. CONCLUSIONS

The results of the implementation demonstrates the suitability of our proposed approach on the field of the digital integration of neural networks. Through the modularization on different architecture levels we keep a system of components which facilitated the construction of different net types conform to the respective problems. By scaling of the weight and interim result vector length we have a additional degree on the optimization of the circuit implementation.

To improve the results furthermore investigations on the impact of the coding scheme with respect to the mapping of the signed digit structures on different target architectures should be done.

5. REFERENCES

[1] C.S.Lindswy and T.Lindblad, “Review of hardware neuronal networks: A users perspective,” http://msia02.msi.se/ lindsey/elba2html/elba2html.html, 1998.

[2] Intel Corporation, *Intel 80170NX Electrically Trainable Analog Neuronal Network, Product Description and Data Sheet*, order no. 290408-002 edition, 1991.

[3] H.P.Graf and D.Henderson, “A reconfigurable cmos neuronal network,” in *Tech. Digest. International Solid State Conference*, 1990, p. 144.

[4] U.Ramacher, “Synapse - a neurocomputer that synthesizes neural algorithms on a parallel systolic engine,” *Journal of Parallel and Distributed Computing*, vol. 14, pp. 306–318, 1992.

[5] Forschungszentrum Karlsruhe (FZK), *SAND/1 Simple Applicable Neural Device Infosheet*, Feb 1997.

[6] IBM France, *ZISC 036 Neurons Users Manual*, 1.2 edition, May 1991.

[7] B.Girau and A.Tisserand, “Area minimization of redundant cordic pipeline architectures,” in *Procceedings of the International Conference on Microelectronics for Neuronal Networks: MicroNeuro96*, Feb. 1996.

[8] M.D.Ercegovac and K.S.Trivedi, “On line algorithms for division and multiplication,” *IEEE Transactions on Computers*, vol. C-26 No 7, pp. 681–687, July 1977.

[9] M.D.Ercegovac, “On-line arithmetic: an overview,” *Real Time Signal Processing VII*, vol. 495, pp. 86–93, 1984.