A New Weight-Programming Structure and Procedure for Pulse-Coupled Neural Networks

BO LIU   JAMES FRENZEL
MRC Institute
University of Idaho
POB 441024, Moscow, ID USA 83844-1024
USA

`http://www.uidaho.edu/~jfrenzel`

Abstract: This paper presents a new method for storing and programming digital weights in a hybrid neural network. The network uses pulse-coupled communication between neurons, compatible with typical CMOS processes, and analog multiplication and addition for modeling neural behavior. Programming of individual neurons is done using existing interconnect, eliminating the need for additional inputs or wiring. Furthermore, the weight storage network supports direct addressing of synaptic weights, allowing the implementation of adaptation and learning. Results from a 4-4-3 array, fabricated in a 1.5 micron process, are presented.

Key-Words: artificial neural networks, pulse-coupled, spiking neurons, weight programming

1 Introduction

Implementation of artificial neural networks in VLSI technology is an active area of research, covering a wide range of approaches [1]. Pulse-coupled neural networks (PCNN) are particularly attractive, because of their ability to mimic biological behaviors and the ease with which pulses can be generated and distributed within an integrated circuit [2, 3].

Our own work focuses on circuits which can be completely configured using "all-digital" weights [4]. Such circuits allow the network to be treated as a programmable device, with the configuration restored or modified as needed. We have shown that these networks are capable of solving a variety of classification problems, as well as implementing Boolean logic expressions [5], and open the possibility of designing recurrent pulse-coupled neural networks using methods from finite automata theory [6].

Configurable hardware implementations often utilize some type of memory storage to store synaptic weights [7, 8, 9]. Centralized memory has the following drawbacks: (1) limited memory ports may prevent all neurons from accessing the weights simultaneously, affecting processing speed; and (2) interconnect between the weight memory and the neural array increases area and reduces the number of neurons which can be implemented. In this paper we present a distributed SRAM structure which avoids both of these limitations by storing the synaptic weights locally and programming them via existing synaptic connections between neural layers. Addressability provides access to individual weights, allowing real-time reconfiguration and adaptation.

The remainder of this paper is organized as follows. In Section 2 we present the neuron circuitry, including the support for weight storage and programming. Section 3 explains a programming algorithm using an on-chip scan chain. Section 4 shows results from a neural array fabricated in a 1.5 μm CMOS process. Finally, our conclusions and areas for future work are discussed in Section 5.

2 Neuron Structure

The structure of the neuron cell is shown in Fig. 1. Multiple synaptic inputs, when active, serve to charge or discharge an internal capacitance, the voltage on which represents the level of excitation. When the voltage on the capacitor reaches a certain value the output of the neuron switches state. Depending upon the specific configuration, this may turn on the feedback transistor and begin discharging of the capacitor. A large NMOS transistor, connected to a global reset signal, is used for initialization.

2.1 Synaptic Inputs

Each synaptic input consists of six transistors, configured using four stored, binary weights, as shown in Fig. 2. W3 determines whether an active input will produce an excitatory or inhibitory effect, while W2–W0 determine the "strength" of the effect. The transistors controlled by W2–W0 are scaled exponentially and together with W3 produce sixteen specific values, corresponding to a weight range of -7 to 7. This circuit represents a 40% reduction in transistor count compared to an earlier design [4], while maintaining the same functionality and advantages of binary weights.

2.2 Output Generation

The voltage on the storage node is sampled using a Schmitt trigger with complementary outputs as a threshold circuit. The "true" output of the circuit drives a feedback transistor. This circuit configuration allows for two types of neuron outputs, "level" or "pulsed", depending upon the size of the feedback device.

If the feedback transistor is smaller than the synaptic input transistors, then active excitatory inputs can source more current than the feedback transistor can sink and the output will remain high as long as the inputs remain active. The time to "fire" and the output high time duration are both functions of the excitatory inputs and their associated weights.

In contrast, if the feedback transistor is larger than the synaptic input transistors, then once the neuron output transitions high the feedback transistor will quickly discharge the storage capacitor and return the output to zero. This, in turn, shuts off the feedback transistor, allowing the excitatory inputs to re-charge the storage node. This behavior represents a pulse-coupled output, with the output frequency dependent upon the synaptic inputs and their weights.

An additional means of generating pulsed behavior is to feed the "true" output back to W3. Under this configuration, active inputs will have an excitatory effect until the neuron "fires," at which point the effect will switch to inhibitory. The transistors in the synapse circuit are sized such that the resultant waveform has a 50% duty cycle.

Regardless of the circuit configuration, active inhibitory inputs will serve to delay the time to fire and reduce the output active time duration, or prevent firing altogether.

2.3 Weight Storage Circuitry

Synaptic weights are stored in 5-transistor SRAM cells, consisting of a write transistor and a pair of cross-coupled inverters. The data input of the SRAM cell is connected to the corresponding synaptic input and the output is connected directly to one of the four synaptic transistors, controlled by W3–W0. A large NMOS transistor, connected to a global reset signal, discharges the storage node prior to weight programming. A multiplexer circuit is provided at the output of internal neurons to enable weight programming of hidden layers, described in the next section. Neurons in the output layer omit this circuit, but may have additional buffers for driving off-chip parasitics. Table 1 provides transistor count and area for the individual components that comprise a neuron. These data correspond to a neuron with five synaptic inputs and four weight bits per synaptic input. The devices and storage capacitor represent 40% of the total neuron area, with weight storage contributing 60%. The remaining neuron area consists of wiring and open area.

3 Neuron Programming

A fully connected array of twelve neurons is shown in Fig. 3, organized into three layers of four neurons, each with four synaptic inputs and a weight range of -7 to 7. Programming of a single SRAM cell involves (a) forcing the corresponding synaptic input to the desired value and (b) asserting the write enable signal. For neurons beyond the input layer, synaptic inputs can be justified via the preceding layer. Assertion of the global reset signal discharges all storage nodes, forcing the Z outputs to '0' and Z' outputs to '1.' The neuron select signals (SEL1–8), connected to the output multiplexers, can then be used to produce the required value. In order to conserve input pins, the values of the select signals are loaded serially into a shift register. The write enable signal is formed from the logical NOR of enable (EN0–EN3) and row select signals (Row_Sel0–Row_Sel3), the combination of which identifies a specific weight bit in a specific row of the neural array. This allows parallel configuration of neurons within a single row.

The procedure is summarized as follows: (1) Assert reset to initiate programming; (2) Reset row and enable counters; (3) Load select bits into shift register; (4) Pulse enable and row select signals low, generating the write enable signal, and loading a single weight bit for each synaptic input in the selected row; (5) Increment the enable counter and repeat steps (3–4) for the remaining weight bits; and (6) Increment row counter and repeat steps (3–5) for the remaining rows.

Each neuron requires sixteen configuration bits, four bits for each synaptic input. Because the weights for all synaptic inputs to neurons in a single row can be loaded in parallel, a single row will be programmed four times, once for each weight bit. However, each iteration may require a new set of select values to be loaded serially, resulting in a total of 128 clock cycles for programming the array in Fig. 3. Parallel scan chains or multiplexed input pins can be used to reduce programming time for large arrays.

4 Results

A neural array, similar to Fig. 3, but with only three neurons in the output layer, was fabricated in the AMIS 1.5 μm CMOS process and is shown in Fig. 4. The connected array consists of 11 neurons, 44 synaptic inputs, and 176 weight storage cells. A stand-alone neuron was included for testing purposes. The total chip area is 2159 $\times$ 2150 μm^2.

Output firing behavior is dependent upon the storage capacitance and the excitatory weights. This array used a poly to poly capacitor with an area of 35 μm $\times$ 35 μm and a capacitance of approximately 1 pF, resulting in an output frequency range of 30–70 MHz. Fig. 5 shows two simulations under different weight settings.

5 Conclusions

We present an improved neuron cell and network structure for weight storage and programming. A programming algorithm is provided, allowing a neural array to be completely configured under external control or from off-chip memory. The design supports multiple pulse-coupled output behaviors, depending upon configuration. A fully connected array of eleven neurons was fabricated in a 1.5 μm CMOS process.

Future work includes circuit improvements to reduce power consumption and the implementation of adaptation algorithms using supervised and unsupervised learning.

Acknowledgment This work was supported by the NSF-Idaho EPSCoR Program and the National Science Foundation under award number EPS-0132626.

References

[1] Murray A., Del Corso D., and Tarassenko L. Pulse-Stream VLSI Neural Networks Mixing Analog and Digital Techniques. *IEEE Trans. Neural Networks*, vol. 2(2), March 1991, pp. 193–204.

[2] Johnson J. and Padgett M. PCNN models and applications. *IEEE Trans. Neural Networks*, vol. 10(3), May 1999, pp. 480–498.

[3] Reyneri L. Theoretical and implementation aspects of pulse streams: an overview. In *Proc. MicroNeuro'99*, Oct. 1999, pp. 78–89.

[4] Liu B. and Frenzel J.F. A CMOS neuron for VLSI circuit implementation of pulsed neural networks. In *Proc. IEEE IECON 2002*, vol. 4. Seville, Spain, Nov. 2002, pp. 3182–3185.

[5] Konduri S. and Frenzel J.F. Non-linearly Separable Cluster Classification: An Application for a Pulse-Coded CMOS Neuron. In *Proc. ANNIE 2003*, vol. 13, Nov. 2003, pp. 63–67.

[6] Konduri S., Frenzel J., and Wells R. Towards a Paradigm for Adaptation in Pulse-Coupled Neural Networks. In *Proc. WSEAS ISPRA'05*. Salzburg, Austria, Feb. 2005. (*to appear)*.

[7] Han G. and Sánchez-Sinencio E. A Flexible and Expendable Neuroimage Processor Architecture. *IEEE Trans. Circuits Syst. I*, vol. 49(9), Sep 1999, pp. 1055–1063.

Table 1: Neuron Components

Component	Devices	Area (μm^2)
Synaptic Inputs	6	$5(50 \times 45)$
Weight Storage	20	$5(4(40 \times 39))$
Schmitt Trigger	7	40×67
Output Mux	6	40×37
Storage Capacitor	0	35×35
Reset & Feedback	2	40×20
NOR gates	16	$4(28 \times 40)$
Total (5 inputs)	161	310×440

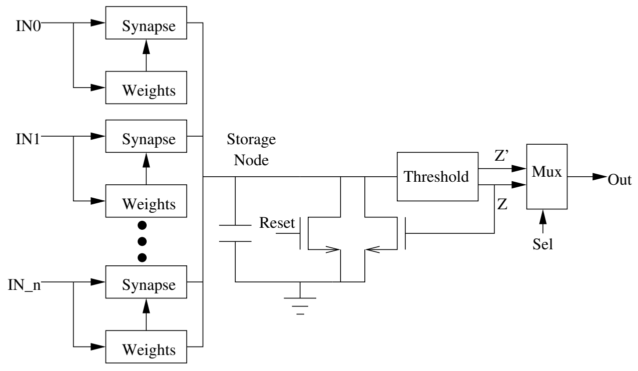

Figure 1: Block diagram of a single neuron with multiple synaptic inputs. Additional signals (not shown) are used for programming the weights, as described in Section 2.3.

[8] Ayala J. *et al.* Design of a pipelined hardware architecture for real-time neural network computations. In *Proc. IEEE MWSCAS-2002*, Aug. 2002, pp. 419–422.

[9] Bracco M. *et al.* Digital implementation of hierarchical vector quantization. *IEEE Trans. Neural Networks*, vol. 14(5), Sep 2003, pp. 1072–1084.

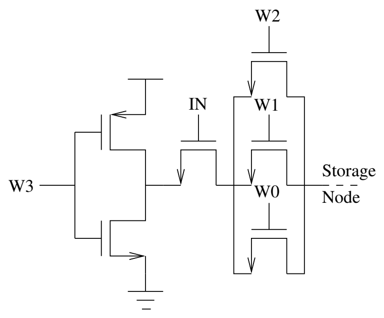

Figure 2: A single synaptic input. Stored weight, W3, controls whether the effect is excitatory or inhibitory, while W2–W0 control the significance of the effect.

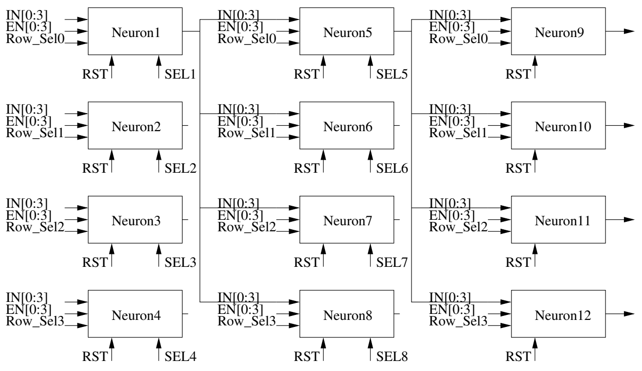

Figure 3: Neural network array. A neuron in one layer connects to every neuron in the next layer. Individual select lines are used to force output values during programming and are not needed in the output layer. Writing of the weight storage is done using a combination of the Row_Sel and EN signals.

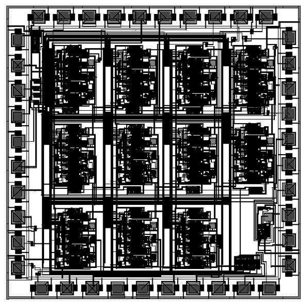

Figure 4: An eleven neuron array, arranged as a 4-4-3 network, fabricated in a 1.5 μm CMOS process.

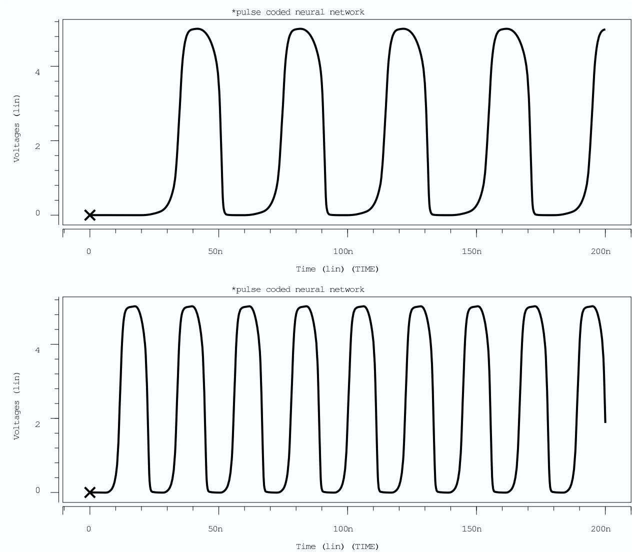

Figure 5: Simulated pulse-coupled output under two weight settings, resulting in an output frequency of 25 or 44 MHz.