Implementation of Pulsed Neural Networks in CMOS VLSI Technology

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

Abstract: This article presents a CMOS implementation of a biologically-inspired neuron. The neuron accommodates multiple excitatory and inhibitory inputs with digital weights and generates a pulse-width modulated output waveform of constant frequency based on the level of activation. The behavior of this implementation is demonstrated and it is shown that combinations of neurons form a complete logic set for realizing Boolean functions. Simulation results using a VHDL model are presented, along with applications for pattern recognition and input encoding.

Key-Words: artificial neural networks, pulse-coded, spiking neurons, VLSI

1 Introduction

Advances in semiconductor technology have led to an explosion of research activity in VLSI circuit implementation of artificial neural networks. Two important efforts of late are [1] and [2]. The work presented in this paper focuses on the implementation of a pulse-coded neuron [3, 4]. Pulse-coded neurons and networks constructed from such are attractive because of their ability to mimic behaviors found in biological neural systems. Furthermore, pulse streams have the advantage that they are easy to generate, possess high noise immunity, and can be distributed over relatively large regions, the latter facilitating the construction of very large neural networks using VLSI circuit technology.

Even within this subcategory there has been a wide range of approaches, ranging from purely analog to all-digital implementations [5]. In [6] and [7], Ota and Wilamowski present a five-transistor voltage-mode neuron. Each neuron accommodates one excitatory and one inhibitory input. Synaptic weight multiplication is realized through sizing of external transistors serving as active coupling resistors.

Chen and Shi use separate circuits to implement synaptic multiplication and summing [8]. The multiplier cell consists of seven transistors and uses an analog voltage to represent the input weight. The input voltage is converted to a current, used to charge a capacitor in an integrate and summing cell. Each summing cell can accommodate multiple synaptic inputs. A sigmoid circuit and voltage-pulse circuit generate the neuron output waveforms. Park et al. employ a similar architecture, but use binary inputs to control transistors serving as active resistors and capacitors, resulting in a programmable RC delay [9].

At the other end of the spectrum, [10] and [11] present architectures suitable for implementation in field-programmable gate arrays. Both designs employ MSI circuits for addition, comparison, and random number generation.

The goal of this work was to develop a neuron with

the following characteristics: (1) utilizes binary signals for controlling the synaptic weights; (2) able to accommodate multiple excitatory and inhibitory inputs; (3) produces a pulsed output, dependent upon the activation level; and (4) suitable for implementation in standard digital CMOS processes.

2 Circuit Implementation

In this design the level of neuron activation is represented by the voltage on a capacitive storage node. In order to ensure that the neuron returns to a relaxed state, even in the absence of inhibitory inputs, a resistive load is attached to this node, forming a leaky integrator. The structure of a complete neuron is shown in Figure 1, with the synaptic input blocks labeled as `Node 1`, etc.

2.1 Neuron Inputs

Figure 2 shows the schematic of a single synaptic input. Each input to the neuron, either from other neurons or fed back from the neuron output, can be set as excitatory, inhibitory, or both, based upon the values of the weights, `WE2-0` and `WI2-0`. Additional inputs to the neuron can be accommodated by replicating this circuit and connecting the output to the storage node.

To realize an excitatory effect, one or more of the weights `WE2-0` must have a logical value of zero. Positive pulses on the input will result in the deposition of charge on the storage node. The PMOS transistors controlled by the excitatory weights are scaled exponentially to provide a total of seven different charging rates. This degree of resolution may be insufficient for certain applications. Excitatory weights of all ones will disable the input from serving in an excitatory capacity. Inhibitory behavior is realized in the same fashion using NMOS devices to discharge the storage node. Because the excitatory and inhibitory weights are independent, it is possible for a single input pulse to cause both an excitatory and inhibitory effect upon the neuron.

2.2 Threshold Circuits

Neuron output behavior should be dependent upon the level of activation. Ideally, the output would become "more active" as the neuron is stimulated in an excitatory manner. This is accomplished using multiple Schmitt triggers with different set points, thus providing a rudimentary approximation of analog to digital conversion. The outputs of the Schmitt triggers forms a binary control word used in generating the pulsed output. Different set points can be achieved through device sizing.

2.3 Pulsed Output Generation

Pulsed output is realized using a ring oscillator with multiple taps. Each tap provides a waveform with a 50% duty cycle and a period equal to twice the delay of the ring. By logically combining selected taps, additional waveforms may be created with different duty cycles but the same period. The binary control word from the threshold logic selects the appropriate waveform based upon the level of activation. At increasing levels of activation, the neuron output switches to waveforms of increasing duty cycle. Neurons receiving this output as an input signal, be it excitatory or inhibitory, will thus experience an increased effect. This is analogous to communicating more than one bit of information to receiving neurons. A control word of all zeroes disables the ring oscillator and forces the output to zero.

2.4 Complexity

Table 1 summarizes the transistor count for the individual blocks in Figure 1. Generation of the pulsed output requires the largest number of devices. The ring oscillator consists of six inverters and a 2-input `NAND` gate, for a total of 16 transistors. Forming and buffering three different waveforms requires ten transistors. Finally, multiplexing these three waveforms and generating the disable signal uses 14 devices.

2.5 Behavior

Spice simulation of a single neuron is shown in Figure 3. This neuron had a single synaptic input and

Table 1: Number of transistors for each component.

Component	# Transistors
Synaptic Input Node (per input)	10
Threshold Circuit (per threshold)	8
Pulse Generation	40

three activation thresholds, requiring a total of 74 transistors. The neuron was stimulated by a waveform with a 15 ns period and a 33% duty cycle. The weights were controlled such that the input was excitatory up until t =100 ns, at which point the input became inhibitory. Close inspection of the neuron output waveform reveals the variation in duty cycle with respect to the voltage on the storage node. Irregular pulses are the result of the multiplexer switching between waveforms produced by the pulse generator circuitry.

2.6 VHDL Modeling

While Spice simulation has been useful for investigating simple neural networks, more complex networks will require a faster simulation model. We have developed a VHDL model of the neuron which utilizes counters and a linear charge rate, dependent upon the weights, to capture the important characteristics of the neuron behavior. Examples of this are shown and compared to the corresponding Spice simulation in the next section.

3 Applications

Given the flexibility of the neuron there are many possible applications. We have investigated several to date. These include implementation of Boolean logic functions, pattern recognition, and input encoding using "Time to First Spike" (TFS).

3.1 Boolean Logic Functions

This section illustrates how the neuron may be used to construct networks that realize Boolean logic functions. From Figures 1 and 2 it is apparent that the neuron performs the equivalent of a logical `OR` operation on the excitatory inputs. If the presence of pulses represents a logical value of '1,' then the behavior of an `OR` gate may be realized by a neuron with two excitatory inputs and the output fed back as an inhibitory input[a]. The latter ensures that the neuron returns to a relaxed state when the excitation ceases, corresponding to a logical value of '0.'

A neuron with one excitatory input and two inhibitory inputs, including the fed back output, will realize the Boolean function $Z = X \cdot \overline{Y}$. Combined with the OR-gate neuron, this can be used to form a network capable of realizing the `XOR` function, as shown in Figure 4. The simulation of this network is shown in Figure 5. The corresponding VHDL simulation appears in Figure 6 and agrees closely with the Spice simulation. A potential concern is the short-circuit current drawn when (1) both excitatory and inhibitory inputs are active and (2) when the voltage on the storage node is near the switching points of the Schmitt triggers. Figure 7 shows a plot of supply current during Spice simulation of the `XOR` gate.

The `XOR` gate simulation demonstrates an important feature, namely that an inhibitory input can "annihilate" or "cancel" the effects of an excitatory input, depending upon the weight settings. This occurs at both neurons N1 and N2 for the input pattern of all ones. The same property can be used to realize an inverter: a signal which is always pulsed whenever inputs are active is used as an excitatory input, and the signal to be inverted is connected as an inhibitory input. When the output of a neuron configured in this manner is connected to the inhibitory input of a neuron such as N1 or N2, the resultant structure realizes the logical `AND` function. The combination of inversion and logical `AND`—or `OR`—forms a complete logic family, allowing the implementation of any combinational Boolean function, given an appropriate network.

Finally, the inhibitory effect may be used to implement an `XOR` function using only two neurons, as shown in Figure 8. In this configuration, neuron N1 computes $a \cdot \overline{b}$. Neuron N2 computes $\overline{a} \cdot b$ and ORs the result with the output of N1 to realize the `XOR`

[a]Weights can be set to render a synaptic input as *solely* excitatory or inhibitory.

operation. In order to produce the correct response, the excitatory input b to N2 must be equal in weight to the inhibitory a input, while the excitatory input from N1 must be stronger. Spice simulation of this structure is shown in Figure 9.

3.2 Hopfield Auto-Associative Memory

Figure 10 shows a two-neuron implementation of a Hopfield auto-associative memory. Training is accomplished by analytically solving for appropriate weight values. Once trained, the memory will retrieve the stored pattern which is closest in Hamming distance to the applied input pattern. Figure 11 shows the Spice simulation of this network with stored patterns, `(0,0)` and `(0,1)`.

3.3 Linear Pattern Recognition

In this application, a single neuron is used to perform linearly-separable pattern recognition. The neuron is configured with five inputs: four inputs corresponding to individual bits of a 4-bit binary word, and a bias input. The neuron is then trained, starting with all weights at zero, to output a '1' or a '0', using the training data shown in Table 2. The distribution of data is shown graphically in Figure 12. Once the network is trained, it will correctly classify patterns from outside the training set. The simulation during training is shown in Figure 13; the weights cease to change around 100 us. Figure 14 shows simulation of the neuron after training, using all possible input patterns.

Table 2: Training Data for linearly-separable pattern recognition. Six patterns out of 2^4 were applied during training.

Set	Input Patterns
'1' Set	(0010), (0101), (1000)
'0' Set	(0011), (0110), (1001)

3.4 Time to First Spike (TFS) Input Encoding

In this application, the input weights and input signal duty cycles determine the elapsed time until the first spike from a resting state. For a single neuron with two inputs, X and Y, the TFS in nanoseconds, T, is determined by

$$T * (W_x \frac{X}{15} + W_y \frac{Y}{15} - \frac{1}{60}) * 0.05 \geq 2.2,$$

where X and Y are integers $[0, 14]$, reflecting the duty cycle of the X and Y input signals, and W_x and W_y are integers $[-7, 7]$, representing the weights. The first two terms in the equation model the charging rates contributed by the X and Y input signals as a linear combination of the respective duty cycle and weight[b]. The third term is a fixed discharge rate modeling the resistive load. The value of 2.2 V represents the trigger point of the first Schmitt trigger. The VHDL simulation of a single neuron with inputs $X = 3$, $Y = 2$, and $W_x = W_y = 1$, is shown in Figure 15. A Spice simulation of the same is shown in Figure 16 for comparison and shows close agreement with the VHDL model.

4 Conclusions

The design and simulation of a pulsed output neuron have been presented. While not the smallest neuron on record, the design does accomplish the original goals. Furthermore, the amount of flexibility given the number of devices compares favorably to other approaches. In particular, the "all digital" nature of the design should facilitate porting to newer, advanced semiconductor processes.

We have demonstrated the capability of implementing Boolean functions and pattern recognition networks. Currently we are designing a test chip for a 1.5 um CMOS process. Future efforts will focus on the following: (1) improving the simulation speed and accuracy of the VHDL model; (2) investigation of effective training mechanisms; (3) design and fabrication of massive neural networks; and (4) identification of additional applications.

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

[b] A negative weight corresponds to an inhibitory input and results in discharging.

References

[1] Rasche C. and Douglas R. Forward- and Backpropagation in a Silicon Dendrite. *IEEE Trans. Neural Networks*, vol. 12(2), March 2001, pp. 386–393.

[2] Diorio C., Hsu D., and Figueroa M. Adaptive CMOS: From Biological Inspiration to Systems-on-a-Chip. *Proceedings of the IEEE*, vol. 90(3), March 2002, pp. 345–357.

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

[4] Reyneri L. Theoretical and implementation aspects of pulse streams: an overview. In *Proc. 7th Intl. Conf. Microelectronics for Neural, Fuzzy and Bio-Inspired Sys. (MicroNeuro'99)*, October 1999.

[5] 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.

[6] Ota Y. and Wilamowski B. Analog Implementation of Pulse-Coupled Neural Networks. *IEEE Trans. Neural Networks*, vol. 10(3), May 1999, pp. 539–544.

[7] Ota Y. and Wilamowski B. CMOS Architecture of Synchronous Pulse-Coupled Neural Networks and its Application to Image Processing. In *Proc. 26th Annual Conf. IEEE Indus. Electron. Soc. (IECON'00)*, October 2000.

[8] Chen L. and Shi B. CMOS PWM VLSI implementation of neural network. In *Proc. IEEE-INNS-ENNS Intl. Conf. Neural Networks (IJCNN 2000)*, vol. 3, 2000.

[9] Park Y., Liaw J.S., Sheu B., and Berger T. Compact VLSI neural network circuit with high-capacity dynamic synapses. In *Proc. IEEE-INNS-ENNS Intl. Conf. Neural Networks (IJCNN 2000)*, vol. 4, 2000.

[10] Hikawa H. Pulse mode multilayer neural network based on floating point number representation. In *Proc. IEEE Intl. Sym. Circuits and Systems (ISCAS 2000)*, vol. 3, 2000.

[11] Maeda Y. and Tada T. FPGA implementation of a pulse density neural network using simultaneous perturbation. In *Proc. IEEE-INNS-ENNS Intl. Conf. Neural Networks (IJCNN 2000)*, vol. 3, 2000.

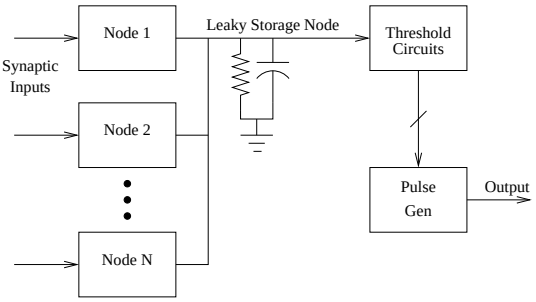

Figure 1: Block diagram of a single neuron with multiple synaptic inputs. Threshold circuits consist of multiple Schmitt triggers, with different set points, forming a digital control word. The pulse generator varies the output duty cycle in response to this control word.

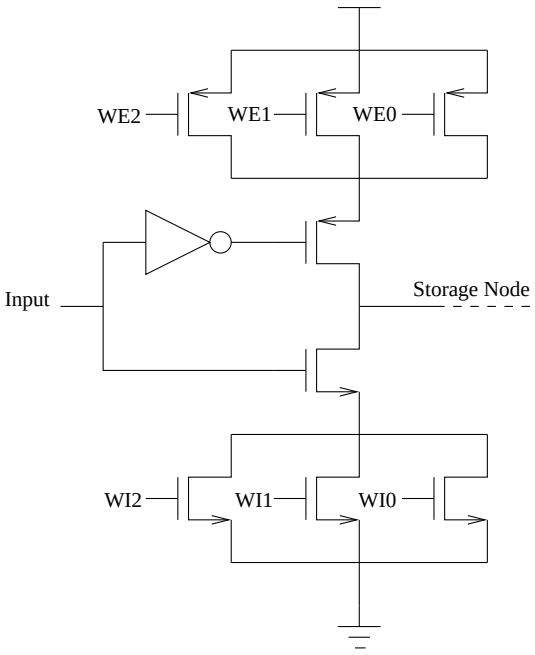

Figure 2: Schematic of a single synaptic input. Transistors labeled `WE2-WE0` represent excitatory weights and are scaled exponentially; `WI2-WI0` are inhibitory weights. The storage node is charge or discharged, based upon the weights, in response to an input high signal.

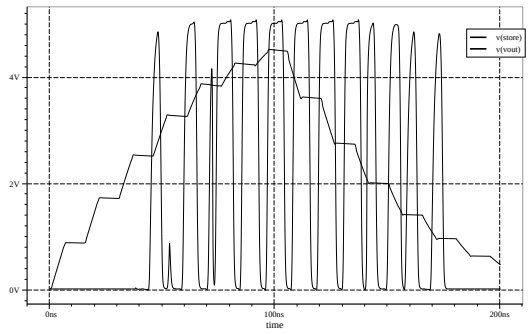

Figure 3: Simulation of a single neuron under excitatory, then inhibitory, stimulus. The two waveforms are the voltage on the storage node, `v(store)`, and the voltage at the neuron output, `v(vout)`.

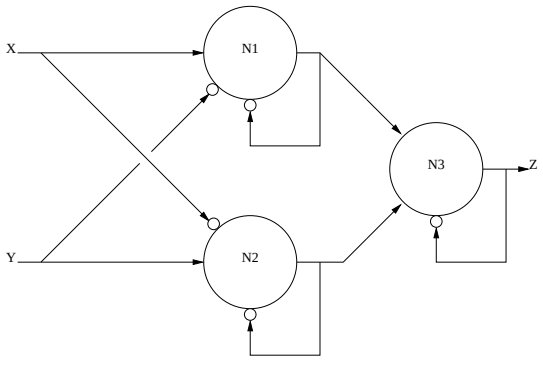

Figure 4: Neural network to realize a Boolean `XOR` gate. Excitatory and inhibitory weights are equal strength, allowing active signals on the inhibitory inputs to "cancel" active signals on the excitatory inputs.

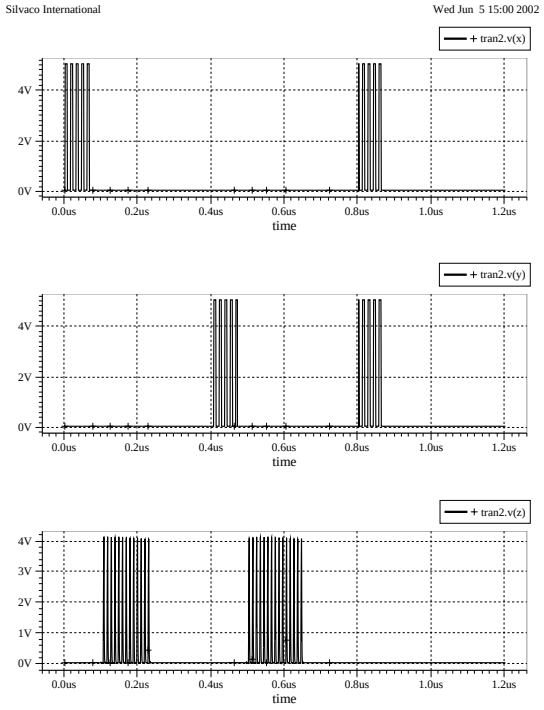

Figure 5: Simulation of the `XOR` gate.

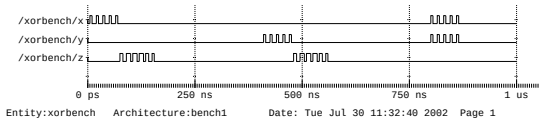

Figure 6: VHDL simulation of the `XOR` gate. The input pattern `(0,0)` is not shown.

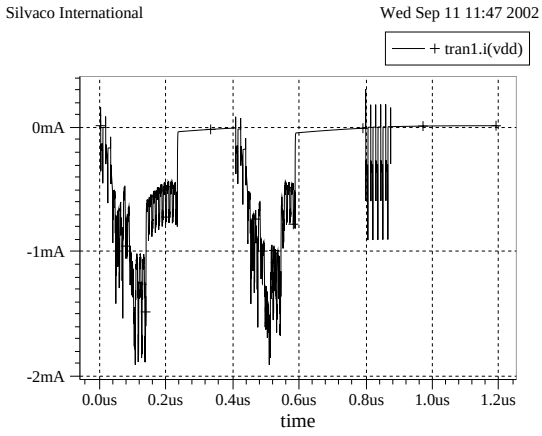

Figure 7: Power supply current during `XOR` simulation. The majority of the power is dissipated by the Schmitt triggers and the ring oscillator when the output is active.

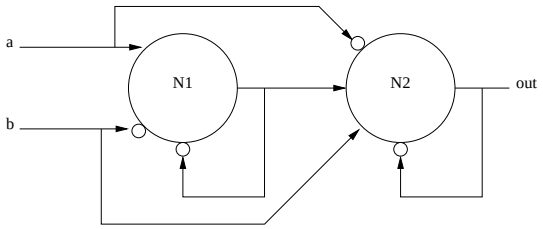

Figure 8: Two-neuron `XOR` gate. Excitatory and inhibitory weights for a and b are equal in strength; the input signal from N1 is stronger than a.

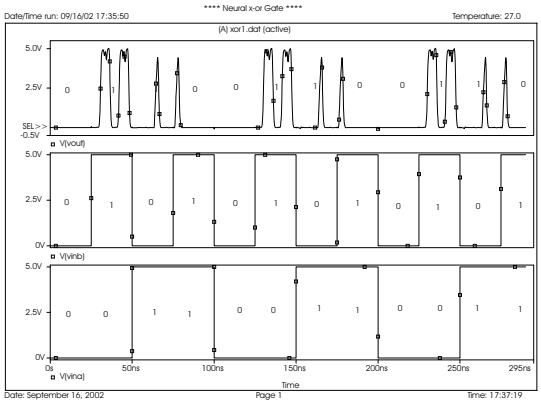

Figure 9: Simulation of the two-neuron `XOR` gate.

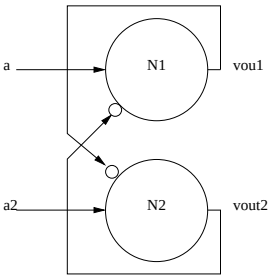

Figure 10: A two-neuron Hopfield network, implementing an auto-associative memory for pattern recognition. The weights (not shown) determine the stored patterns.

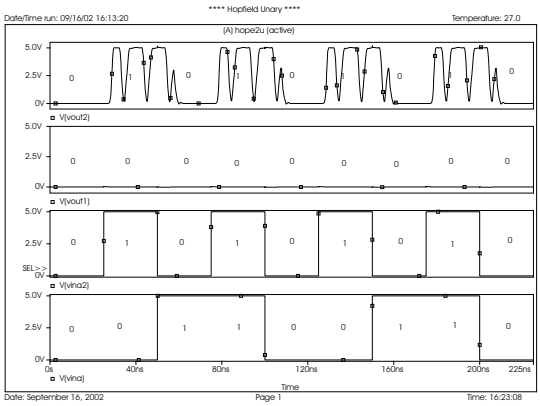

Figure 11: Simulation of the Hopfield network. The weights were solved analytically to store the patterns `(0,0)` and `(0,1)`.

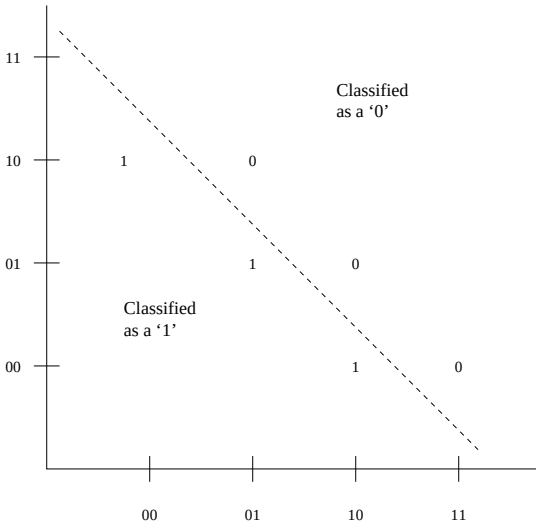

Figure 12: Distribution of training data. Vertical axis marks the leftmost two bits from the input patterns and the horizontal axis is the rightmost two bits. Once trained, the network will appropriately classify patterns from outside the training data.

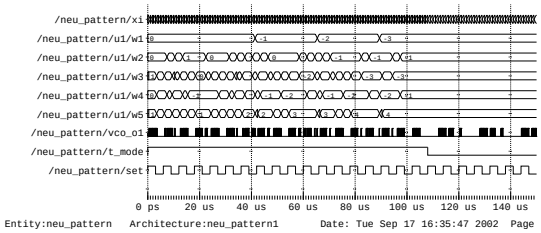

Figure 13: VHDL simulation of linear learning during training. The signal `t_mode` is used to enable training and the signal `set` represents the desired output for specific patterns from the training set, labeled `xi`. Note that the weights have stopped changing by 100 us.

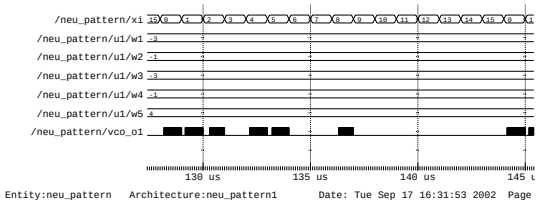

Figure 14: VHDL simulation of linear pattern recognition after training. All possible input patterns are now applied to the network.

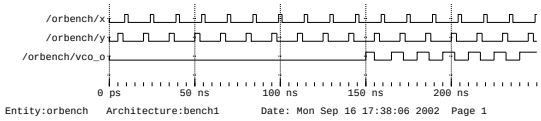

Figure 15: VHDL simulation of TFS encoding. The duty cycle of $x = 3/15$ and $y = 2/15$. The weights, W_x and W_y are both set to one.

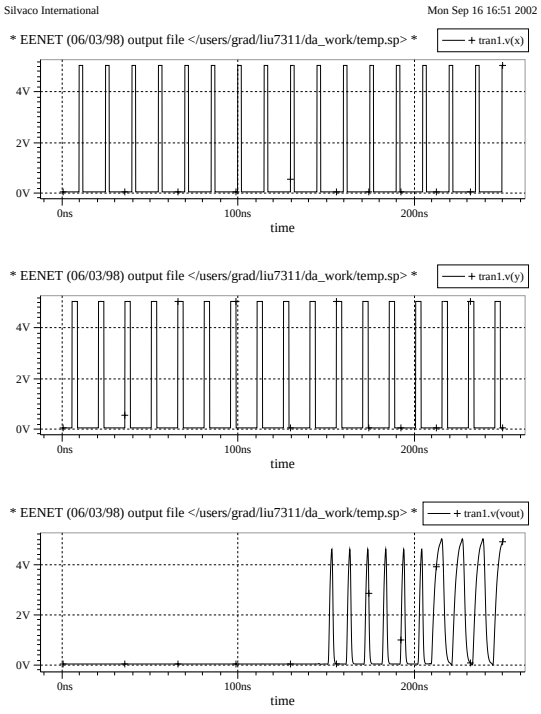

Figure 16: Spice simulation of TFS encoding.