REVIEW OF HARDWARE pRAMs

T G Clarkson, C K Ng, C Christodoulou and J Bean
Department of Electronic and Electrical Engineering
King's College London, Strand
London WC2R 2LS

Abstract

This paper reviews the development of the probabilistic RAM (pRAM) from its conception to the present day. One of the chief aims of this model of an artificial neuron has been to implement suitable learning rules in hardware. This work spans the development of single neuron models to an architecture for implementing large-scale pRAM networks and on to artificial neurons with temporal responses.

pRAM development

Four generations of pRAM hardware have so far been developed, three of which are in VLSI [1,2,3]. In 1988, the first hardware pRAMs were built as 2-input devices constructed (Fig. 1) using LSI logic parts. A small net, comprising two of these devices was shown to give results which agreed well with earlier theoretical analysis. The results of this net show the distinctive pulse-train output of such devices (Fig. 1). Tasks such as mutual inhibition or mutual excitation were successfully learned. The training was achieved by running the pRAM hardware for a single iteration on a PC interface card and the output of the pRAMs was then read. A reinforcement learning rule was applied in software on the PC and the updated weights were written back to the hardware before the next iteration was started.

Following the success of this small-scale system the VLSI design of a 4-input pRAM was completed in early 1990. These devices still relied upon a host computer to perform the learning algorithm and are in the category of "off-chip" learning devices. Even if only a

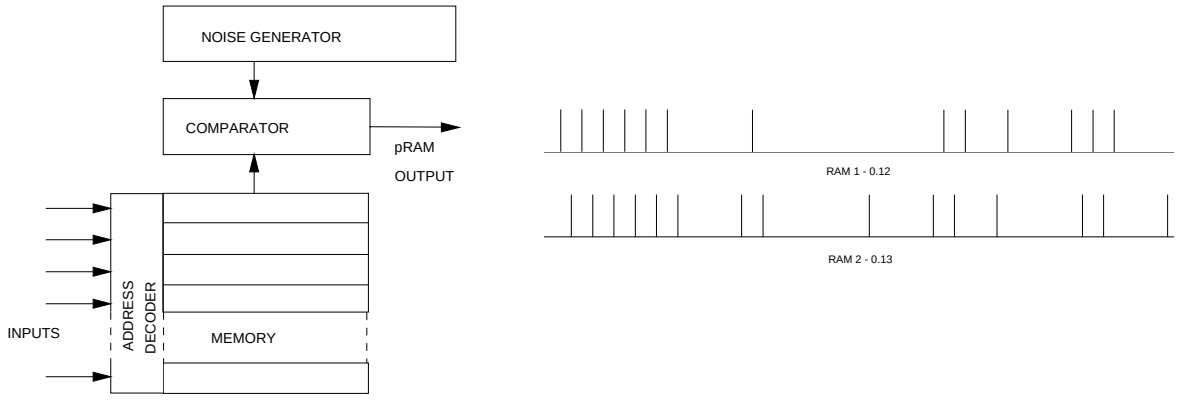

Figure 1. The 2-pRAM and two resultant output traces of a network showing the mean firing frequencies in the displayed windows and overall.

modest sized network of such devices were built, then the interconnection requirements would be inconvenient and any reconfiguring of the network could only be achieved by rewiring. It was necessary to provide each pRAM neuron in a network with its own noise generator to avoid unwanted correlations between neurons. This was achieved by the external selection of one of sixteen maximal length sequences; each pRAM was configured to generate a different sequence. There are clearly limits to this approach if large-scale nets are envisaged.

To overcome these limitations, the third-generation pRAM design employed serial processing of pRAMs, on-chip reinforcement learning and a reconfigurable network architecture. Other features of this form of architecture are described in Section 3. The number of inputs per pRAM was fixed at 4, but 256 such learning pRAMs were incorporated on the same chip. A chip and its local RAM memory comprised a pRAM module. A decision had to be made early on in the design process whether to use local Hebbian learning or global reinforcement training in the design; local learning was chosen, as this was of major interest at the time.

By the time that this set of pRAM modules was fabricated, our attention had turned to global learning and a number of image processing applications were being investigated. For these reasons, a further development was required, leading to the fourth generation of pRAMs which allows a number of learning techniques to be used, all of which are based on reinforcement training. This has been achieved by allowing the reward and penalty inputs for each neuron to be reconfigurable by a look-up table. For example, a reward input to a neuron may either be connected to external pins on the chip which receive a global reward or penalty signal, or to the output of another pRAM. The number of inputs per pRAM was increased to six, since in image processing it is often useful to process a central pixel and its four nearest neighbours requiring a 5-input pRAM. However, when local learning is used, the output of a neuron is used in the determination of its own reward or penalty signal; this is achieved using *auxiliary* pRAMs which take the input vector (5 bits, for the reason given above) and the output of the learning pRAM (1 bit) as input. Therefore, at least 6 inputs per neuron are required to support the use of *auxiliary* pRAMs. The pRAMs within a chip are made to be uniform so that they may be configured to be either learning pRAMs or *auxiliary* pRAMs for local learning, as required. Thus pRAM modules now contain 256, 6-input pRAMs.

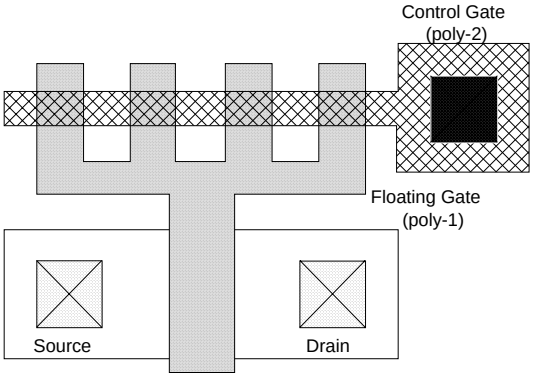

Figure 2. A floating gate transistor (Lee, Sheu and Yang).

Analogue pRAMs

The provision of RAM and a noise generator per neuron demands a significant amount of digital circuitry. The possibility of the use of analogue processing to replace these two main components is being explored. Owing to the limited accuracy of analogue weights (approximately 6-bits) and the ease of fabricating and testing digital circuitry, digital pRAMs were first developed. However, the

functional component parts have been designed in VLSI analogue form and their characteristics are being measured with a view to building and evaluating analogue pRAMs in the future.

The three functional blocks which comprise a pRAM neuron are the memory, comparator and noise source (Fig. 1). These have been fabricated on the same chip, but are not connected so that each component may be separately tested and their performance measured. We require well-defined weights in the EEPROM cells which are trainable within a small error margin and the noise source must have a uniform amplitude probability across its range but must be uncorrelated with other pRAMs in a net.

Analogue EEPROM Cells

Each word of the digital RAM can be replaced by a transistor whose drain current represents the stored probability. The temporary nature of analogue dynamic RAM is unsuitable for this application. Near permanent analogue memory is available in EEPROMs and have been reported using standard CMOS technologies [4]. They are at present not a standard technological item. Devices similar to those of [4] have been fabricated in a standard reduced 3μ process and are being evaluated (Fig. 2). They exhibit the same characteristics as purpose designed EEPROMs, where the write cycle is relatively long and elevated voltages are required. The non-linear and approximate relationship between applied analogue input voltage and output current is acceptable in this application, because the learning algorithm can compensate for this.

Comparator

An analogue comparator is a standard part but in this application, it is desirable to have minimum hysteresis or a very low threshold. Any threshold effect will distort the effective probability distribution of the noise source. It should be noted that in digital circuits a comparator is a simple function but the analogue equivalent requires careful consideration.

Analogue Noise Source

The Probability Density Function (PDF) of the noise generator must be constant (some small deviation is acceptable) over a given range of amplitudes and zero elsewhere. Naturally occurring noise generators are Gaussian and would require significant processing and amplification to provide useful signals. Two sampled analogue circuits have been investigated as possible noise sources. In the first a triangular wave generator which has the required PDF is used. If it is randomly sampled, the resulting output is then a noise signal with a suitable PDF. When there are many uncorrelated triangular wave based noise generators on a chip, frequency locking has to be avoided by carefully choosing a variety of oscillator frequencies in adjacent neurons' generators. The alternative noise generator is derived from studies of non-linear dynamics in digital systems [5][6]. It uses a chaos generator based on a non-linear amplifier which has a tent-shaped transfer function. Its sampled and delayed output is fed back to its input, Fig 3. It

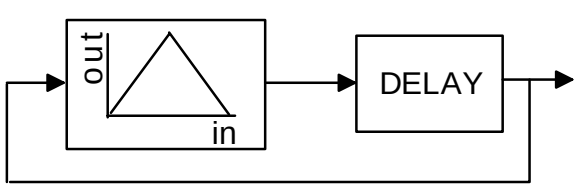

Figure 3 The chaos generator

provides a noise- like output signal. If the slopes of the transfer function are linear, the PDF is constant over the permitted input range. Characteristics of circuits built using this principle are being investigated. Studies will be undertaken to ensure independent

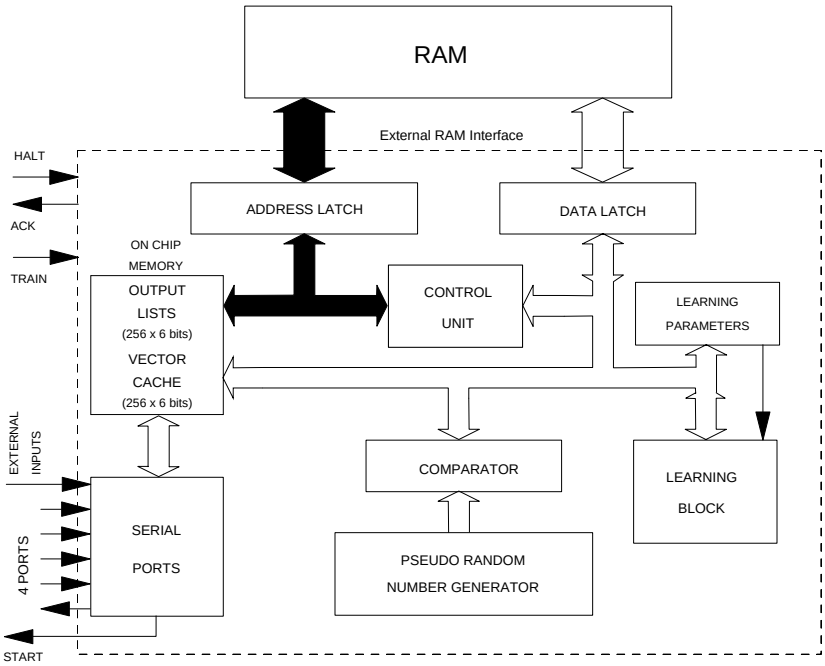

Figure 4 The 4th generation pRAM modular architecture.

operation of adjacent generators.

Digital pRAM Modules

For the fourth generation pRAM designs, there are 256 digital 6-pRAM learning neurons per package (Fig. 4). In the prototypes for this design, one custom integrated circuit and two external RAM devices comprise a module. It would be advantageous to include all RAM on-chip eventually. The connectivity of the pRAM inputs is reconfigurable through a lookup table and the reward-penalty signals are now reconfigurable so that local or competitive learning can be achieved on-chip. Expansion of a net beyond 256 neurons is achieved through the use of four serial links which enable one pRAM module to communicate with up to four neighbours. Instead of using dedicated inputs for the external (state) inputs, a single pin is now used so that these inputs may be entered serially. This not only reduces the pin count, but also allows up to 256 external inputs to be used, if required. This input is now identical to the existing four serial links and may be used as such if required, which allows a local cluster of up to 1536 pRAMs, in 6-pRAM modules, to be constructed.

Interface to a host computer

This module can operate autonomously, but if supervision by a host processor is required, the chip may be halted using the HALT input and the address and data buses used by the host processor when ACK is asserted. In this way, the weight memory may be read or written to or the connection pointer table may be rewritten, so reconfiguring the network.

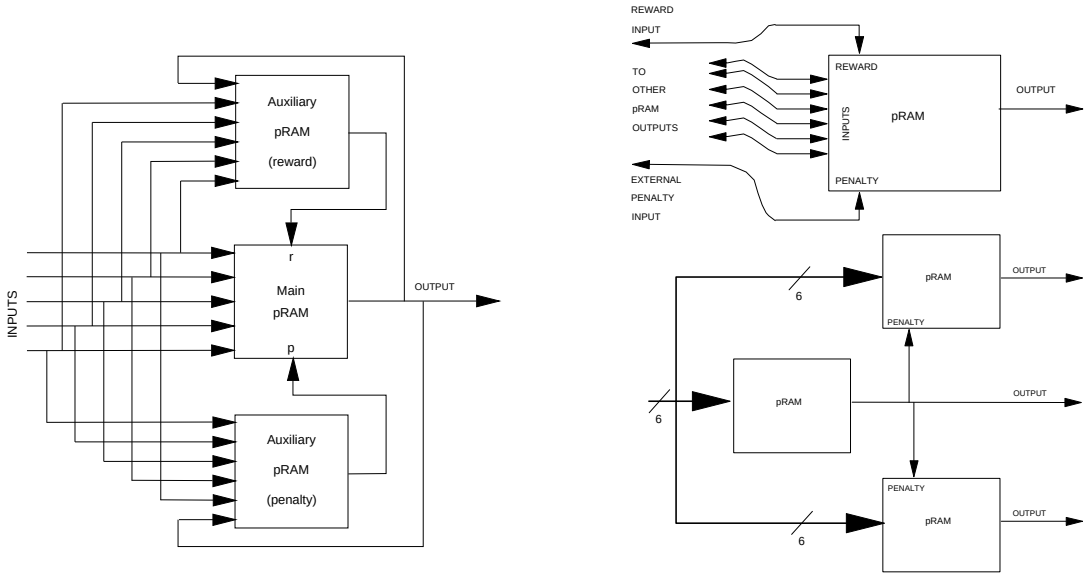

Figure 5 left: local learning, upper: global reinforcement and lower: competitive learning

The TRAIN input is used to enable or disable training. When training is disabled, the memory update cycle is cancelled and the processing time is thereby reduced.

On-chip Training

The learning rate and decay rate used in the training algorithm are stored in registers, which may be written to using the memory port. In this way, the learning rate can be changed as the learning process proceeds.

Three examples of training configurations are shown in Figure 5. Fig.5 (left) shows how two pRAMs are used as *auxiliary* pRAMs to a central, learning pRAM. In this case, only 5 of the 6 inputs of the learning pRAM may be used, the remaining input is permanently connected to either 0 or 1 to keep it inactive. The *auxiliary* pRAMs are non-learning and are loaded with memory weights according to the kind of actions that the learning pRAM is required to reinforce or penalise. This kind of learning is unsupervised. Each learning pRAM of this kind within a pRAM module may have a distinct behaviour which it is required to reinforce or penalise.

Fig. 5 (upper) shows how global learning is achieved. All pRAMs which are trained in this way have their reward inputs connected to a single external reward signal. This global reward signal will have been determined by an external environment as part of a supervised learning scheme. The penalty inputs are similarly connected to a global penalty input. The reward and penalty inputs are independent in this architecture and also in our reinforcement learning rule.

Fig. 5 (lower) shows how the central pRAM, when firing, may be used to penalise two other pRAMs to achieve a form of competitive learning. Where groups of neurons form mutually inhibiting or mutually exciting clusters, the activities of neighbouring pRAMs are combined, using further pRAMs, into a single reward or penalty input.

Since each pRAM in a module is uncommitted, it may be used as a learning pRAM or an *auxiliary* pRAM as desired. The pRAM module may be used to build a single, multi-layer

network or a number of smaller, independent networks. The reward and penalty inputs are likewise uncommitted so that different learning methods can be employed within the same module.

Order of processing pRAMs

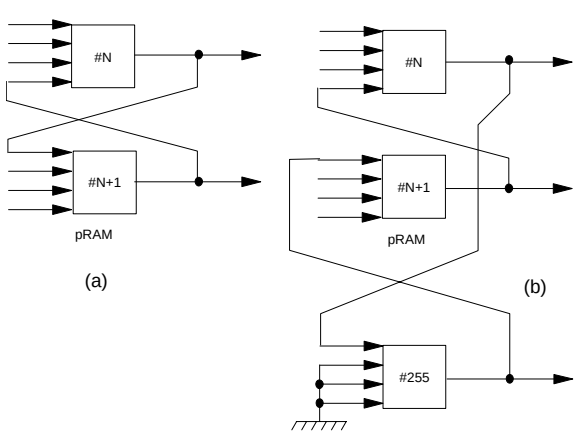

Figure 6 The use of a dummy pRAM

Since pRAMs within a module are serially processed, and since these pRAMs are logically interconnected, the third generation pRAM modules stored both the previous state and the new state of each pRAM so that for the current pass all pRAM inputs looked at the previous state table, because the new state table changed throughout the pass. At the end of each processing pass, the new output state table was copied to the previous state table. These devices also performed the local learning process as each of the 256 pRAMs was processed. This means that for a *N* layer network, N passes of the pRAM module are required in order to propagate a new input state through to the output. This is acceptable for local learning as the input vector and the output state are known at the time that training is applied.

For global reinforcement learning, the reward and penalty signals fed back to the net at time *t*, must coincide with the internal states of the network which caused the output at time *t*, so that the correct internal states or correlations are rewarded or penalised. This would be hard to achieve if the net has internal states representing t, t-1 and t-2 at the time that reinforcement is applied.

For the fourth generation pRAM modules, therefore, the whole network is processed, from input to output, within one pass of a module. This implies that pRAMs in the input layer should have the lowest pRAM numbers and pRAMs at the output should have the highest pRAM numbers, i.e. they are processed last. This applies even if the network is spread over multiple pRAM modules, since all modules operate concurrently. No "previous state" table is now used and a network designer must be aware that the output states of pRAMs are changing. It is usually a straightforward process to set up the connection tables for the pRAMs to ensure that no pRAM depends upon the output at *t-1* of another pRAM of lower number; for regular pyramidal structures, this is always so. However, if connections span layers, or two pRAMs are cross-coupled (Fig.6a), it is not possible to meet this condition.

A means of overcoming this potential problem of pRAM processing order has been found by the use of *dummy* pRAMs. A pRAM with a high number is used as a *dummy* pRAM. By definition, such pRAMs are processed last in any pass. This pRAM is used to latch the previous state of any lower-numbered pRAM which cannot meet the ordering restrictions above. It therefore retains this state throughout the next pass, until it is itself updated. This is shown in Fig. 6b.

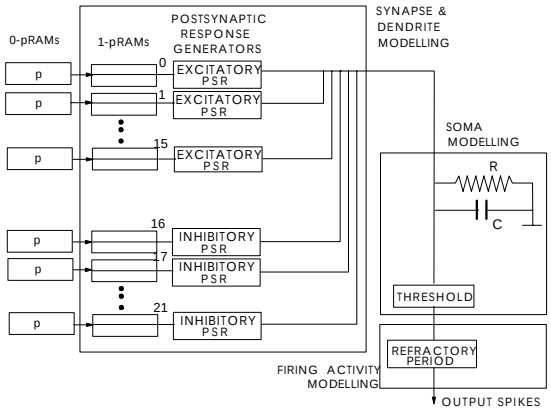

Figure 7. Analogue hardware outline of the TNLI neuron model

Temporal pRAMs

A digital, hardware-realisable version of a neuron based on the noisy-leaky integrator model has been investigated. This model is different in architecture from the devices previously described in this paper, but it uses pRAMs at the synapses and so is also discussed here.

An analogue hardware outline of this model, which we call Temporal Noisy-Leaky Integrator [7-9], is shown in Fig. 7. The model uses a pRAM at each input and a Hodgin and Huxley [10] equivalent circuit for the leaky cell membrane. The 0-pRAMs shown at the input level produce random spike trains of controlled mean spike frequency according to the given probability p and are not part of the model, but are shown to indicate how real-valued inputs are converted to spike trains. The 1-pRAMs that follow model the stochastic neurotransmitter release [11] by the synapses of real neurons. Neurophysiological evidence [12,13] indicates that there are a large number of synapses each having a probability of releasing quanta of neurotransmitter. The probability distribution of quantal release depends on the number of nerve impulses arriving at the synapse and also on the spontaneous activity of the cell body where there is no nerve impulse. This noisy nature of the synapses is perfectly matched with the stochastic pRAM behaviour (α_0 being the spontaneous probability and α_1 being the stochastic probability in the 1-pRAMS). In other words, the use of the pRAMs enables the generation of noise at the synaptic level of the TNLI which complies with the biological neuron, as opposed to other models in which noise is generated at the threshold level. Generalisation is also improved by this noise injection [14]. In addition, the 1-pRAMs at the input level of the TNLI increase slightly the irregularity of the effective input spike trains.

This can be detected from the effect of the pRAMs on the Coefficient of Variation of the spike trains which is given by: $C_V=\sqrt{(\sigma^2_{\Delta t}/\Delta t_M)}$ where t_M is the mean interspike interval and $\sigma^2_{\Delta t}$ is the variance of the spike train given by:

$$\sigma^2_{\Delta t} = \frac{1}{N}\sum_{i=1}^{N}(\Delta t_i - \Delta t_M)^2$$

where N is the number of spike intervals and Δt_i is the interspike interval.

For a random Poisson process C_V=1 & the t_i histogram follows an exponential shape. In the TNLI, where discrete time steps exist due to the pRAMs:$C_V(1) = \sqrt{(1\text{-}p)}$ for the input spike trains produced by the 0-pRAMs with firing probability p and $C_V(2) = \sqrt{(1\text{-}pq)}$ for the effective input

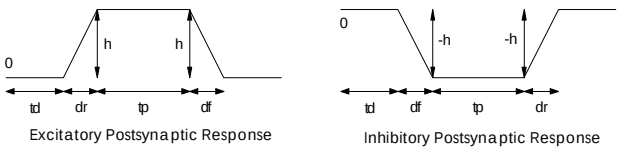

Figure 8. Shapes of the Postsynaptic Responses used in the TNLI

spike trains (i.e., the ones inducing the postsynaptic response currents), if $\alpha_0 = 0$ and $\alpha_1 = q$ as memory contents of the 1-pRAMs.

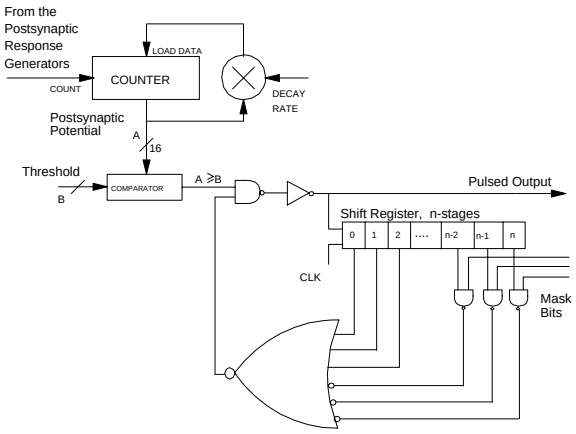

Figure 9. Digital hardware structure of the TNLI.

The postsynaptic temporal response current generators (PSR) shown in the diagram of Fig. 8, model the dendritic propagation of the postsynaptic potential. For every spike generated by the pRAMs, the PSR generators produce postsynaptic current responses ($PSR_{ij}(t)$) of controlled shapes, shown in Fig. 8, which can either be excitatory or inhibitory. These particular ramp shapes were chosen for the postsynaptic responses (instead of smooth exponential ones), due to the fact that they can easily be implemented in hardware and their defined parameters can be trained. In addition, these shapes result in smoother responses after passing through the leaky integrator circuit, if long rise and fall times (d_r and d_f) are selected, compared to responses produced by rectangular shapes commonly used as inputs to neurons. This enables us to reproduce the smooth postsynaptic potentials produced in distal dendrites of real neurons [15]. The postsynaptic current responses are summed temporally and the total postsynaptic current response is fed into the RC circuit (Fig. 7). The capacitance C and the resistance R represent the soma and the leaky membrane of real neurons respectively and therefore this circuit models the decay that occurs in the somatic potential of the biological neuron due to its membrane leak. The capacitance C and the resistance R are fixed at a suitable value to give the leaky membrane time constant. This intrinsic leakage of R is used to give additional temporality of a biologically realistic form. Finally, if the potential of the capacitor exceeds a constant threshold (V_{th}), the TNLI neuron fires. It then waits for a refractory period (t_R) and fires again if the potential is above the threshold. Therefore, the maximum firing rate of the TNLI is given by $1/t_R$.

In the digital hardware structure of the TNLI, the pRAMS at each of the TNLI inputs will take the form of a probabilistic RAM controller with a serial update digital VLSI structure [3]. An iterative procedure is used to fetch each pRAM from the external memory and a postsynaptic response generator attached to each pRAM produces the required postsynaptic shape. The parameter values governing this shape will be determined by programmable registers which model the postsynaptic current response. These postsynaptic current responses are accumulated in the counter where they are multiplied at regular intervals by a decay rate (Fig. 9). The decayed synaptic potential is routed back to the counter via a load input. This digitally produces the exponential RC-decay. The decayed postsynaptic response has a threshold applied and if it exceeds that, the TNLI neuron fires according to the refractory period conditions [16]. The threshold is implemented using a comparator and a shift register and gate circuit is used to inhibit firing while in the refractory period (see Fig. 9).

As mentioned above, in the TNLI we may incorporate hyperpolarising inhibition with negative current pulses of controlled shape as shown in Fig. 8. Such responses are produced by certain Postsynaptic Response generators which are assigned to be inhibitory ones. In practice, this is achieved by programming a negative value for *h* in a register (Fig 8). The number of inhibitory generators is thus variable. In order to demonstrate the effects of inhibition, the number of the inhibitory postsynaptic response generators was varied and the change in the relationship between the Mean Input Current (I_M) and the output frequency of the TNLI was observed. I_M in the TNLI neuron *i* is given by:

$$I_M = \sum_{j=0}^{N} f_j \times PSR_{ij}^{*} \quad (1)$$

where f_j is the mean input spike frequency which in our simulations is the same for each input *j* and PSR_{ij}^{*} is the time integral of the postsynaptic current (PSR_{ij}) produced by a spike arriving on input line *j*. N is the total number of input lines (or pRAMs). At the TNLI inputs, random spike trains of controlled mean frequency (f_j) were utilised (produced by the 0-pRAMs) which were unaffected by the 1-pRAM action since their memory contents were set to '1' for an input spike and '0' for no spike and thus they fired for each input spike.

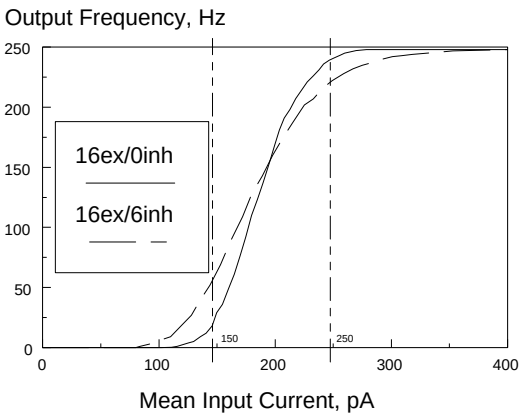

Figure 10. The effect of inhibition on the TNLI output transfer

Results were taken with 16 excitatory PSR generators from two runs with 0 and 6 inhibitory PSRs. In order to obtain the same Mean Input Current we had to increase the mean input frequency (f_j), while the number of inhibitory inputs was increased. The output characteristic of the TNLI for the two configurations above is shown in Fig 10. First it was observed that the TNLI gives a sigmoidal non-linear transfer function instead of a step function. This behaviour seems to be similar to that of the formal neuron which has a sigmoid transfer function given by: $y = 1/(1 + \exp(-\alpha A_i))$ where α is a constant that determines the slope of the sigmoid and A_i is given by: $\Sigma x_j w_{ij}$ where x_j is the jth input to neuron *i* and w_{ij} is the connection weight value from neuron *j* to neuron *i*. A_i is equivalent to I_M in the TNLI (eqn. 1). Fig. 10 shows that the introduction of the inhibition has the same effect as decreasing the value of α in the sigmoid whereas in formal neurons inhibition only affects A_i.

From Fig. 10 it can be observed that the output frequency with inhibition is higher for low Mean Input Current (I_M) values and lower for high I_M values compared to the no inhibition case. In order to explain this, two snapshots of the Membrane Postsynaptic Potential were taken for I_M = 150pA and I_M = 250pA (vertical lines on Fig. 10), for the two extreme cases of 16ex/0inh and 16ex/6inh. The snapshots showed that the inhibition produces more fluctuations on the membrane potential though it does not change its measured mean saturation level. Therefore, in the case of low I_M, where the mean saturation level of the membrane potential is below the threshold, the membrane potential of the 16ex/6inh case is able to exceed the threshold more frequently than in the 16ex/0inh case due to the

fluctuations and thus give a higher output frequency. However, in the case of high I_M the mean saturation level of the membrane potential is above the threshold and so in the 16ex/6inh case, due to the high fluctuations again, the membrane potential is able to go below the threshold more frequently than in the 16ex/0inh case and thus give a lower output frequency. This explains the reduced slope of the sigmoidal characteristic curves of Fig. 10 in the presence of inhibition.

In conclusion, inhibition in the TNLI not only reduces the mean input current for the same mean input frequency, but it also modifies the transfer function of the neuron, by increasing the fluctuations of the input current around its mean saturation value. This goes beyond the assumption underlying the formal neuron used in Artificial Neural Networks where it is assumed that positive and negative inputs add linearly and then pass through a fixed sigmoidal transfer function, whereas in the TNLI the sigmoidal transfer function is modified by the signals passing through it. The effect of the fluctuations cannot be established in experimental neurobiological observations [9], since the inputs to the biological neuron cannot be controlled. Therefore, the TNLI, despite its simplicity, can be useful for modelling and understanding real neuron behaviour.

Conclusion

The major thrust of our research has been not only to develop hardware-realisable neurons, but to develop models which allow learning processes to be performed in hardware also.
A number of learning algorithms have been proposed for the pRAM but the easiest to implement is the reward/penalty (or reinforcement) algorithm since this algorithm always produces weights in the range [0,1]. This also has the benefit of operating on a single spike output, rather than requiring a pulse train in order to calculate the weight update.
This gives the advantage of stochasticity in the learning process which is useful to a net in that the net may better explore its environment and it is less likely to be trapped in local minima during training.

Both the pRAM neuron and the reinforcement training may be mathematically analysed [2] and a proof of convergence exists for the learning rule.

The pRAM modules are currently being used for image processing, pattern classification and real-valued function learning. Only by the use of such hardware is it possible to classify objects in a video image within one frame time of 20ms.

A family of pRAM structures is envisaged for the future, which will include learning, non-learning and temporal devices, thus enabling pRAM modules to be applied to a wide range of tasks.

References

1. T G Clarkson, D Gorse and J G Taylor (1989), "Hardware-realisable models of neural processing", Proceedings of IEE International Conference on Artificial Neural Networks, 242-246, London.
2. T G Clarkson, D Gorse, J G Taylor, C K Ng (1992), "Learning Probabilistic RAM Nets Using VLSI Structures", IEEE Transactions on Computers, Vol 41, **12,** 1552-1561.

3. T G Clarkson, C K Ng, C Christodoulou, Y Guan (1993), "The pRAM: An Adaptive VLSI Chip", IEEE Transactions on Neural Networks, *Special Issue on Neural Network Hardware*, (to appear in May 1993).
4. B W Lee, B J Sheu and H Yang (1991), "Analog Floating Gate Synapses for General Purpose VLSI Neural Computation", IEEE Trans. Circuits and Systems, Vol 38, No 6, 654-658.
5. T.Kilias, Eindimensional Zeitdiscrete Chaotische Abbildungen, PhD thesis, Technische Universität Dresden, 1992.
6. A.C. Davies, Relating Pseudorandom Signal Generators to Chaos in Non-linear Dynamic Systems, Accepted for ECCTD '93 DAVOS, King's College London, 1993.
7. Christodoulou C, Taylor J G, Clarkson T G & Gorse D (1992). The Noisy-Leaky Integrator model implemented using pRAMs. *Proceedings of the Int. Joint Conf. in Neural Networks 1992*, Baltimore, **Vol. I,** 178-183.
8. Christodoulou C, Bugmann G, Taylor J G and Clarkson T G (1992). An extension of the Temporal Noisy-Leaky Integrator neuron and its potential applications. *Proceedings of the IJCNN '92*, Beijing, **Vol. III,** 165-170.
9. Christodoulou C, Bugmann G, Clarkson T G and Taylor J G (1993). The Temporal Noisy-Leaky Integrator with additional inhibitory inputs. *New Trends in Neural Computation*, Eds. Mira, Cabestany and Prieto, Springer-Verlag, 465-470.
10. Hodgkin A L and Huxley A F (1952). A quantitative description of membrane current and its application to conduction and excitation in a nerve. *Journal of Physiology* (London) **117,** 500-544.
11. Gorse D, Taylor J G (1991). A continuous Input RAM-Based Stochastic Neural Model. *Neural Networks*, **Vol**. **4**, 657-665.
12. Katz, B. (1969). *The release of Neural Transmitter substance*. Liverpool University Press, Liverpool.
13. Raymund Y. K. Pun, Elaine A. Neale, Peter B. Cuthrie, and Philip G. Nelson (1986). Active and Inactive Central Synapses in the Cell Culture. *Journal of Neurophysiology*, **Vol. 56,** No. 5, 1242-1256, USA.
14. Guan Y, Clarkson T G, Taylor J G & Gorse D (1992). The application of noisy reward/penalty learning to pyramidal pRAM structures. *Proceedings of the IJCNN'92*, Baltimore, **Vol. III**, 660-665.
15. Shepherd G. M. (1990). *The Synaptic Organisation of the Brain*, (3rd edition), Oxford University Press.
16. Bressloff P C and Taylor J G (1991). Discrete Time Leaky Integrator Network With Synaptic Noise. *Neural Networks*, **Vol. 4**, 789-801.