# A Low-Power VLSI Arrhythmia Classifier

Philip H. W. Leong, *Member, IEEE,* and Marwan A. Jabri, *Senior Member, IEEE*

***Abstract*—The design, implementation, and operation of a low-power multilayer perceptron chip (Kakadu) in the framework of a cardiac arrhythmia classification system is presented in this paper. This classifier, called MATIC, makes timing decisions using a decision tree, and a neural network is used to identify heartbeats with abnormal morphologies. This classifier was designed to be suitable for use in implantable devices and a VLSI (very large scale integration) neural-network chip (Kakadu) was designed so that the computationally expensive neural-network algorithm can be implemented with low power consumption. Kakadu implements a (10, 6, 4) perceptron and has a typical power consumption of tens of microwatts. When used with the arrhythmia classification system, the chip can operate with an average power consumption of less than 25 nW.**

## I. Introduction

An implantable cardioverter defibrillator (ICD) is a device which monitors the heart and delivers electrical shock therapy in the event of a life-threatening arrhythmia. Central to the success of such a device is the reliability of the arrhythmia classification algorithm.

Current ICD's perform classification with the main criterion being the heart rate measured from a single lead. These methods have been found to be unreliable in their classification of many common arrhythmias, often causing inappropriate therapy to be delivered [1]. Most of the incorrect classifications made by current devices are caused by the overlap of heart rate ranges between the various arrhythmias.

Any classifier used in an ICD must also be implementable with very low power consumption. This is because battery life inside an ICD must be of the order of five years and the classifier must consume as little power as possible. Although many excellent algorithms for arrhythmia classification already exist, they are usually computationally expensive and hence are not suitable for use in implantable devices.

This paper describes the implementation of a hybrid decision tree/neural-network algorithm for classifying arrhythmias. This algorithm, called MATIC, has been shown to have improved performance over existing algorithms [2], however, neural networks are very computationally expensive. To achieve low power consumption, an analog subthreshold VLSI (very large scale integration) implementation of the neural network was made. The described architecture enables one to keep the benefits of a powerful morphology classification algorithm yet maintain low power consumption.

Manuscript received May 18, 1993; revised December 6, 1993 and June 24, 1995. This project was supported by Australian Generic Technology Grant 16029 and Telectronics Pacing Systems Ltd., Australia.

The authors are with the Systems Engineering and Design Automation Laboratory (SEDAL), Department of Electrical Engineering, University of Sydney 2006, Australia.

IEEE Log Number 9415040.

Section II describes the neural network VLSI chip (Kakadu) which was designed to perform the morphology part of the MATIC system. A comparison of various training algorithms and their ability to train the Kakadu chip is made in Section III. In Section IV, examples of Kakadu applied to simple classification problems are described. Section V describes the MATIC algorithm and the system's performance when applied to a large database of arrhythmias. Finally, a brief discussion followed by conclusions on this work are presented.

## II. Kakadu MLP Chip

The Kakadu chip was implemented using low-power analog techniques because they have the following advantages:

- Subthreshold analog circuits enable implementations which consume very little energy,
- They are easily interfaced to the analog signals in an ICD (in contrast to digital systems which require analog to digital conversion),
- Analog circuits are generally small in area,
- Fully parallel implementations are possible, and
- A certain amount of fault tolerance may be exhibited by the neural network.

### A. Architecture

Kakadu implements an artificial neural network based on the multilayer perceptron model [3]. A block diagram of the chip is shown in Fig. 1. The chip takes voltage inputs and passes them through the first array of synapses to produce six pairs of hidden layer currents. These currents are converted to voltages using linear resistors that are external to the chip. The same nodes are used as voltage inputs to the next layer which produces output currents which are converted to voltage outputs by the third neuron layer.

The main blocks of the chip are two synapse arrays, a current source, and weight addressing circuitry. The synapse's digital-to-analog converters are binary weighted current sources which are controlled by digitally stored weights. A common current source is used to supply bias voltages to synapses in each digital-to-analog converter (DAC). The circuit can be operated over a wide range of bias currents.

Although inputs to the neural network are analog, synapse values are written digitally. This enables configuration of the chip to be performed digitally but keeps the actual signal processing in the analog domain. The synapse array appears as an 84–word RAM (the first layer having $10 \times 6$ words and the second layer having $6 \times 4$ words) with a six-bit word size. Synapses are addressed by row and column through pairs of multiplexed row and column shift registers.

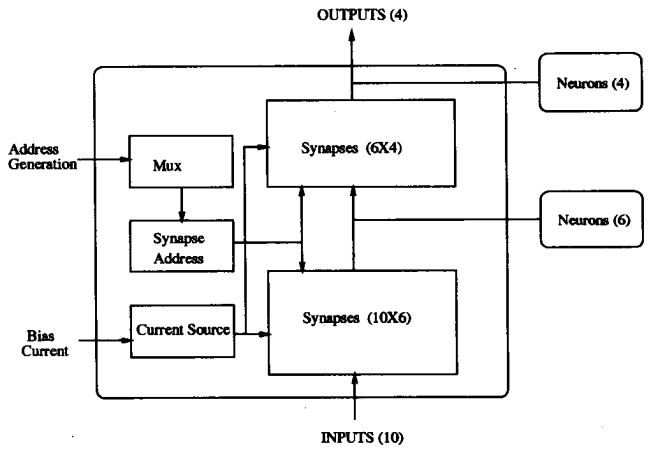



Fig. 1. Block diagram of the Kakadu MLP chip.

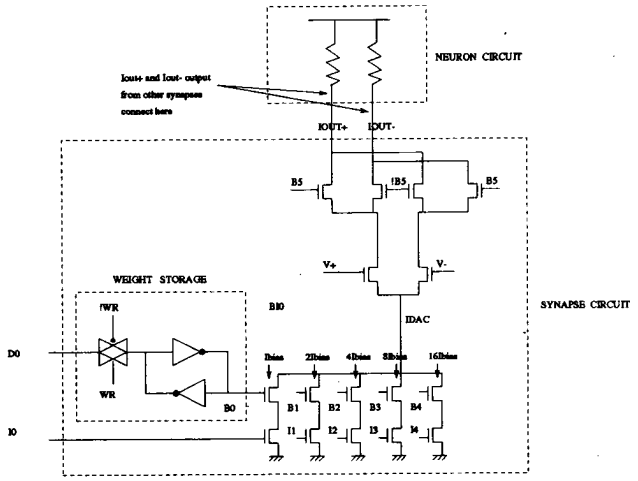



Fig. 3. Synapse and neuron circuitry.

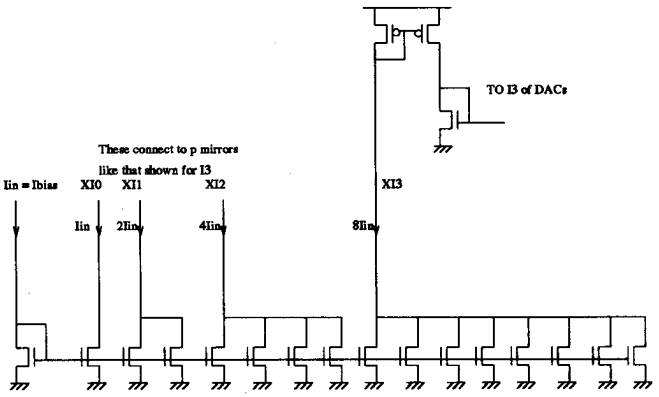



Fig. 2. Current source circuitry (four bits shown). Typical bias currents used are 6.63 nA, and all of the $n$–transistors on the bottom of the circuit are the same size so $I = I_{\text{in}}$.

### B. Implementation

*Current Source:* A single current source is used to provide biases for all synapses of the chip. The current source (Fig. 2) is constructed by summing unit current sources. For transistors with uncorrelated matching properties, summing $N$ unit current sources improves the matching by a factor of $\sqrt{N}$ [4]. Correlated matching properties such as changes in doping or oxide thickness are addressed by arranging the current sources in a common centroid configuration [4]. Large (553 $\mu$m$^2$) transistors are used for the current source although smaller (81 $\mu$m$^2$) transistors are used inside the DAC's to keep the total synapse area small.

The bias current is controlled by an off-chip current or voltage. Since all of the currents feeding the synapses are derived from this single input, the entire circuit can be switched off by making $I_{\text{in}}$ equal to zero. The current source can operate in either strong inversion or subthreshold mode, depending on the magnitude of the bias current. In the experiments, subthreshold operation was used.

*Synapse:* The synapse is composed from registers which store the weight values, a linear DAC, and a transconductance multiplier. The bias current is the same as the unit current for the DAC so each DAC can output $\pm 31$ times the bias current. A circuit diagram of the synapse is shown in Fig. 3.

Since synapses are the most numerous elements in a neural network, the size of the network that will fit in a given area is controlled by their dimensions. Although small synapses are required, the matching of crucial transistors (the five mirror transistors connected to I0–I4) within the synapse is proportional to the square root of the transistor area and so these transistors should be made as large as possible. A compromise was reached in selecting 81 $\mu$m$^2$ transistors for the I0 to I4 mirrors within the synapse.

Storage of the synapse values is achieved using registers, the value of which are converted to analog values via a DAC. To achieve a small synapse area, the registers were designed to be as narrow as possible since each register contains six flip-flops.

The DAC is constructed through current summing. Each bit of the DAC is controlled by a pass transistor which can be turned on or off depending on the value stored in the (static) input flip-flop (B0–B4). I0–I4 are voltages taken from the current source which serve to bias the currents in powers of two. B5 is used to encode the sign and is included in the synapse rather than the DAC. The entire synapse array appears as a large (write only) register to the controlling digital circuitry which programs the weight values.

The DAC is connected to a transconductance multiplier to form a synapse. The multiplier has a pair of voltage inputs, a current input (from the DAC), and a pair of current outputs. The transfer function of this multiplier is given by the relation

$$I_{\text{out}+} - I_{\text{out}-} = \begin{cases} +I_{\text{DAC}} \tanh\left(\frac{\kappa(V_+ - V_-)}{2}\right) & \text{if B5} = 1 \\ -I_{\text{DAC}} \tanh\left(\frac{\kappa(V_+ - V_-)}{2}\right) & \text{if B5} = 0. \end{cases} \quad (1)$$

The multiplier is linear with the current inputs (from the DAC) and nonlinear to the neuron voltage inputs. This is the desired situation as if they were reversed, the tanh function would only serve to compress the range of weight values available and would not allow nonlinear problems to be solved. The DAC only produces positive values. Current switching logic controlled by B5 enables the output to be changed in

sign if a negative weight is desired. The $V_+$ and $V_-$ inputs are from either neurons or input pins. Output of the multiplier are two current sinks.

The area of a synapse in 1.2 $\mu$m double metal, single poly nwell technology is 106 × 113 $\mu$m which includes all of the weight storage registers and switches, I0–I4 current mirrors, multiplier, and sign switching circuitry. A neural network can be constructed from a single current source (described in Section II-B) and a synapse array. A larger, single layer version of Kakadu has been designed, which contains a $50 \times 50$ array of synapses, current source, and weight addressing logic on a 7.2 × 7.2 mm die.

*Neurons:* In a low-power system, where the neuron input current can be of the order of 10 nanoamps (nA), a high impedance of the order of 1 MΩ is required. This is hard to implement in standard metal-oxide semiconductor (MOS) technology because diffusion and polysilicon do not have the high resistance necessary, and an active circuit with the desired transfer characteristic is difficult to design. If on-chip neurons are used, a method of measuring the activation of at least the output neurons is required for training, and this requires buffers to drive the signals off chip.

A possible solution to this problem is to implement the neurons using off-chip resistors. The resistors are provided off chip to allow easy control of the impedance and transfer characteristics of the neuron. The neurons also serve as test points for the chip. It is envisaged that these neurons will later be integrated onto the chip using either an active circuit or a high resistance material such as that used in a static ram process. Since the neurons are implemented externally to the chip, nonlinear neurons can also be used.

A resistor, however, has a linear characteristic which, at first glance, appears unsuitable. This problem was addressed by implementing the nonlinear characteristic required by the neural network in the synapse instead of the neuron. Using this technique, the nonlinearity of the Gilbert multiplier is used to an advantage.

The linear neurons mean that the transfer function of the Kakadu network are proportional to that of the synapses alone, and the transfer function is

$$u_i = \sum_{j=1}^{N_l} w_{ij} f_l(a_j) \tag{2}$$

$$a_i = \alpha u_i \tag{3}$$

where $u_i$ is the summed output of the synapses, $a_i$ is the neuron output, $\kappa$ and $\alpha$ are constants, $l$ denotes the $l$th layer $(0 \leq l \leq L-1)$, $L$ is the total number of layers (namely 2), $N_l$ is the number of neuron units at the $l$th level $i$ is the neuron number $(1 \leq i \leq N_l)$, and $f_l(x) = \tanh(\frac{\kappa x}{2})$.

For a two-layer network, (3) is very similar to the typical multilayer perceptron model as illustrated in Fig. 4. Any set of inputs can be scaled so that they are within the linear range of $f_l(x)$ and so the initial nonlinearity applied by $f_0(x)$ (i.e., $f_l(x)$ where $l = 0$) does not change the computational capability of the circuit. There is an absence of a nonlinearity in the final layer, and this can be thought of as a linear output unit. Equation (3) can thus be rewritten in the familiar multilayer perceptron form

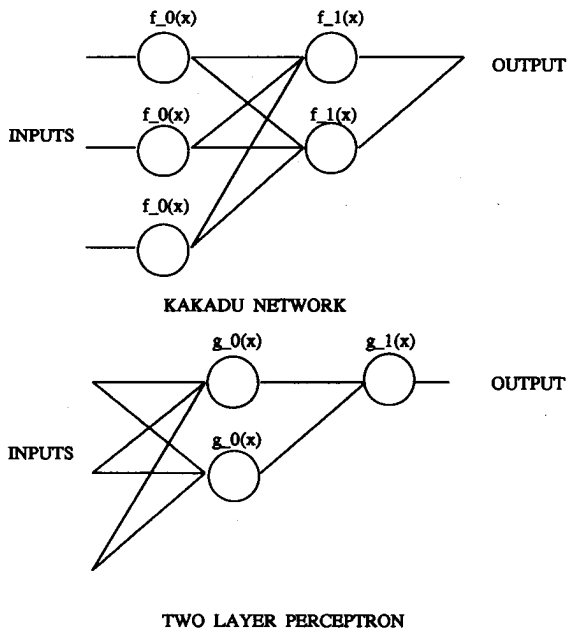



Fig. 4. Comparison between the Kakadu architecture and a multilayer perceptron. The nonlinearity is depicted by the circles and lines represent synapses.

$$u_i = \sum_{j=1}^{N_l} w_{ij} a_j \tag{4}$$

$$a_i = g_l(u_i) \tag{5}$$

where $g_0(x) = \alpha \tanh(\frac{\kappa x}{2})$, $g_1(x) = \alpha x$, and it is assumed that the inputs have been initially passed through $g_0(x)$.

As shown in Section IV, this does not affect the neural network's ability to solve highly nonlinear problems such as exclusive or (XOR) and the parity problems. The disadvantages of using off-chip neurons are that since the currents must travel through pins so pin leakage may affect the circuit and also, for larger networks, the number of pins required may become excessive. The larger parasitic capacitances associated with the neurons also reduce the bandwidth and possibly increase the power consumption of the system. It should also be noted that all analog VLSI neural network implementations have limited input and output ranges since they are (at best) bound by voltage and current restrictions imposed by the supplies.

*Kakadu MLP Chip:* Kakadu was fabricated using Orbit Semiconductor's 1.2 $\mu$m double metal, single poly nwell process. A photomicrograph showing the main synapse blocks, row shift registers, and the current source is shown in Fig. 5. Kakadu has 10 input, six hidden, and four output neurons and hence is called a (10, 6, 4) neural network. It can implement any smaller network than (10, 6, 4) by setting unused synapse values to zero. A summary of the major chip features is shown in Table I.

### C. Chip Testing

*Jiggle Chip Tester:* The Kakadu chip was tested using the "Jiggle" test jig [5]. Jiggle was designed at the University

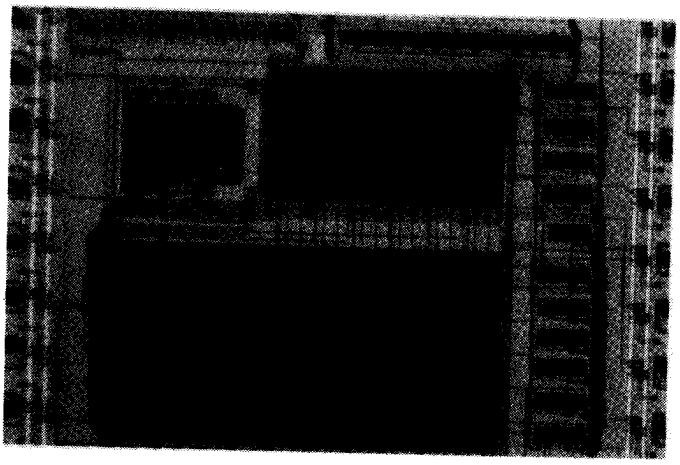


Fig. 5. Photomicrograph of Kakadu MLP chip.

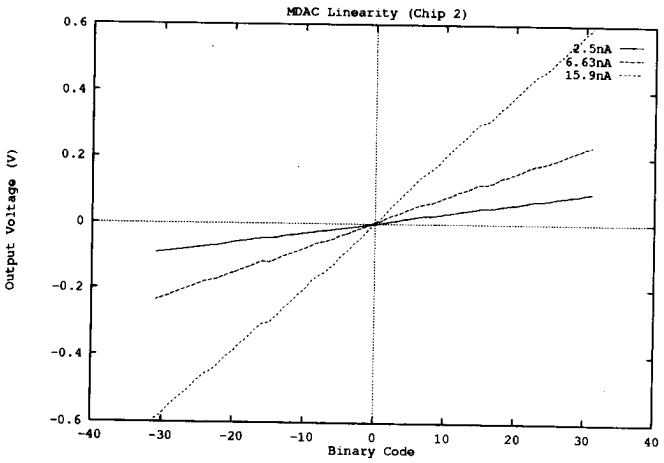



Fig. 6. MDAC linearity test (Chip 2).

TABLE I
KAKADU MLP CHIP SUMMARY

| | |
|---|---|
| Technology | $1.2\mu m$ double metal, single poly nwell |
| Chip Size | $2.2 \times 2.2$ mm |
| Synapses | $84 \times 6$ bit MDAC |
| Power Supply | 3V |
| Power Consumption | $25\mu W$ (typical) |
| Power Consumption | $25nW$ (duty-cycled) |
| Rise Time (0.2V swing into 2pF) | $30\mu s$ |

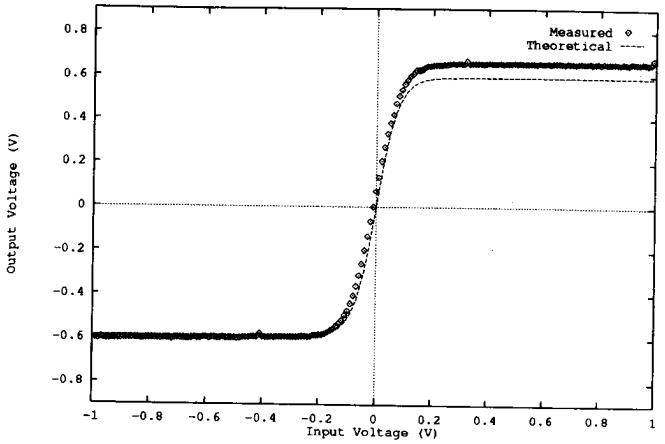



Fig. 7. Synapse transfer function.

of Sydney, Australia, and is a general purpose chip tester having 64 12-bit analog input–output channels as well as 64 digital input–output channels. Jiggle connects to a VME bus, and the versa module eurocard (VME) cage is interfaced to a Sun SPARCstation IPC via a Bit3 Model 466 SBUS to VME converter.

Jiggle allows arbitrary analog or digital signals to be presented to the pins of the test chip and thus allows software control of the weight updating and training of the Kakadu chip. For the experiments described below, a supply voltage of 3 V, a bias current of 6.63 nA, and neuron values of 1.2 MΩ were used.

*MDAC Linearity Test:* The transconductance multiplier used in the Kakadu MDAC has a transfer function described by (1). The output of the DAC ($I_{\text{DAC}}$ in Fig. 3) is equal to $I_{\text{out}+} - I_{\text{out}-}$ (1) and so for fixed input voltages the equation can be simplified to

$$I_{\text{out}} = \beta I_{\text{DAC}} \tag{6}$$

where $\beta$ is a constant and

$$I_{\text{DAC}} = \begin{cases} +\sum_{k=0}^{4} 2^k B_i & \text{if B5} = 1 \\ -\sum_{k=0}^{4} 2^k B_i & \text{if B5} = 0. \end{cases} \tag{7}$$

$I_{\text{out}}$ is connected to a neuron (a 1.2 M pullup resistor), and so the output can be measured as a voltage. Plots of the (measured) MDAC linearity for three bias currents are shown in Fig. 6. One can see that monotonicity is not achieved for the bias current of 6.63 nA (at which the chip is operated). The two points at which it is not monotonic are when the absolute value of the DAC input changes from 15 to 16 and is hence due to the most significant bit current source. At 15.9 nA, however, the DAC is monotonic. Training is used to account for the nonlinearity of the DAC.

*Synapse Transfer Function:* The synapse followed by a neuron has a transfer function described by

$$V_{\text{out}} = R(I_{\text{out}+} - I_{\text{out}-}) \tag{8}$$

where $R = 1.2 \times 10^6$ and $(I_{\text{out}+} - I_{\text{out}-})$ is given by (1). A curve fit was used to find $\kappa$ (26.0719), and a plot of the measured and expected synapse transfer function can be seen in Fig. 7.

*Power Consumption:* It is useful to be able to estimate the power consumption of a chip like Kakadu. This is a function which is linear with the weight values since $I_{\text{DAC}}$ in Fig. 3 is the current drawn for that particular synapse. A number of current consumption measurements were made for different weight values and then a least-squares fit was used to derive

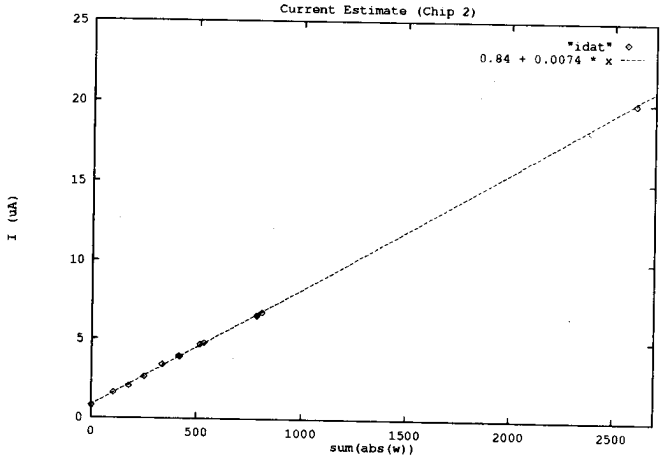



Fig. 8. Current consumption curve fit (bias current = 6.63 nA).

the current consumption formula

$$I_{\text{KAKADU}} = 0.842 + 0.00736 \sum_{i=0}^{N} |w_i| \quad (\mu\text{A}) \qquad (9)$$

where $w_i$ is the $i$th weight, $i$ indexes through all of the weights in the chip, and $I_{\text{KAKADU}}$ is the current consumption in $\mu$A.

Fig. 8 shows the measured current dissipation of the chip and the curve fit of (9) to this data. Note that the maximum power consumption (60 $\mu$W) of this chip occurs when all the weights are set to the maximum value.

## III. Comparison of Training Algorithms

Training of analog neural-network chips is much harder to achieve than their digital counterparts. Mismatch of transistors, nonperfect transistor models, and noise means that a mathematical formula for the neural-network transfer function cannot be attained, and thus a formula for the gradient also cannot be reliably computed.

The training problem can be posed as an optimization problem in which a function, $f(w)$ can be defined as

$$f(w) = \sum_{p} (ff(w,p) - o(p))^2 \qquad (10)$$

where $p$ = input patterns, $ff$ = Kakadu feedforward function, $w$ = weights, $o(p)$ = desired output for pattern $p$, and the optimization task is to minimize $f(w)$.

Training is achieved by using the chip in feedforward mode, with the chip interfaced to a computer via Jiggle. An initial set of weights are written to the chip, and the training vectors are applied in turn to compute $f(w)$. The weights are updated by the training rule to minimize $f(w)$, and this procedure repeated until the error reaches a sufficiently low value. In all of the training experiments, the error termination condition was that the mean-squared error divided by the number of training patterns must be less than $1 \times 10^{-4}$.

TABLE II
Training Algorithm Comparison (Batch Mode). All Numbers of the Form $(m, s)$ Are the Mean and Standard Deviation Values

| Algorithm | Iterations | Training Error | Converged |
|---|---|---|---|
| bp | (1.27e+03, 5.60e+02) | (1.42e-03, 1.98e-03) | 14 |
| cprs | (1.63e+03, 3.48e+02) | (7.85e-04, 6.78e-04) | 12 |
| csa | (3.52e+01, 1.84e+01) | (3.36e-04, 4.79e-05) | 20 |
| sa-only | (1.52e+02, 8.27e+01) | (3.08e-04, 8.22e-05) | 19 |
| sa | (8.83e+01, 8.10e+01) | (3.51e-04, 5.30e-05) | 20 |
| sed | (2.00e+03, 0.00e+00) | (1.45e-03, 6.13e-04) | 0 |
| swnp | (1.34e+03, 3.45e+02) | (4.32e-04, 2.36e-04) | 18 |
| wp | (4.66e+02, 1.62e+02) | (3.58e-04, 2.61e-05) | 20 |

Several techniques were used to train Kakadu by minimizing (10). A comparison between different training algorithms is given below. An extensive set of experiments were performed to assess the capabilities of the Kakadu architecture in terms of training and generalization performance, and to compare the speed of various training algorithms. The algorithms used were:

- BP — Backpropagation. The gradient calculation were derived from (3).
- CPRS — Constant perturbation random sign [6].
- CSA — Combined search algorithm as described in Section IV-1.
- SA-only — Pure simulated annealing.
- SA — Combined search algorithm with simulated annealing instead of the partial random search.
- SED — Stochastic error descent [7].
- SWNP — Summed weight neuron perturbation [8].
- WP — Weight perturbation [9].

A complete description of the training experiments has been reported [10], and a summary of the relevant results are presented here. Note that WP is sequential with respect to the weights, whereas SWNP, SED, and CPRS are parallel weight perturbation methods.

The training problem was that of ICEG morphology classification. The training set consists of eight QRS complexes taken from a single patient (four VT 1:1 and four NSR). The testing set consists of 220 patterns (150 VT 1:1 and 70 NSR). Each experiment was repeated 20 times with different starting conditions.

### A. Batch Training

Tables II and III show the performance of batch mode training, where weight updates were computed for each pattern and applied when all patterns have been processed. The maximum number of iterations was set to 2000. The results show that in batch mode, Kakadu can be trained to produce very good generalization performance.

### B. On-line Training

Tables IV and V show the performance of the training in online mode, where weight updates are computed for a given input pattern and then applied before the consideration of the next pattern. The maximum number of iterations was set to

TABLE III
TESTING COMPARISON (BATCH MODE). ALL NUMBERS OF THE FORM $(m, s)$ ARE THE MEAN AND STANDARD DEVIATION VALUES

| Algorithm | Testing Error | Correct (%) |
|---|---|---|
| bp | (1.18e-03, 1.39e-03) | (82, 28) |
| cprs | (5.55e-04, 7.61e-04) | (89, 15) |
| csa | (3.28e-04, 3.93e-04) | (96, 6) |
| sa-only | (7.46e-04, 5.91e-04) | (91, 8) |
| sa | (1.08e-03, 1.06e-03) | (87, 13) |
| sed | (9.13e-04, 6.69e-04) | (81, 15) |
| swnp | (2.51e-04, 1.69e-04) | (96, 4) |
| wp | (3.80e-04, 1.69e-04) | (95, 3) |

be 1000. The combined search algorithms (CSA, SA, and SA-only) do not appear in the tables since they work only in batch mode. As can be seen from the table, all algorithms are capable of training Kakadu with very good generalization performance.

### C. Discussion on Training

All the training algorithms tried were successful in training Kakadu in batch or online mode, or both. The algorithm that was most successful in both batch and online mode was sequential WP. None of the online methods managed to converge every time.

Of the batch algorithms CSA, SA-only, SA, and WP converged in all cases. Of these four algorithms that converged every time, CSA had the best generalization performance.

Since only six bit weights are used, the limited range and resolution of the weights make (10) discontinuous and hence difficult to train. These training results show, however, that reliable training and generalization can be achieved despite this problem.

## IV. OTHER CHIP TRAINING EXAMPLES

Although Kakadu was designed primarily for low-power arrhythmia classification, it has been applied to other classification problems. In this section, a series of training examples of increasing complexity are described.

An output was considered correct if the difference between the measured and desired output was less than a particular margin. In all of these experiments, this margin was set to be 0.08 V.

There is a linear relationship between the value of the neuron gains (which are determined by the resistance of the neurons) and the value of the bias current. For example, the Kakadu chip will produce the same output if the neuron gain is doubled and the bias current halved. Since the neuron gains were fixed at 1.2 MΩ, the bias current can be adjusted to suit the problem. This was not necessary in the experiments that were performed, but it is envisaged that this may be necessary for some problems. In all of the experiments below, the bias current used was 6.63 nA unless otherwise stated.

The combined search algorithm [11] was chosen to perform the training in all of the Kakadu training experiments because of its reliable training and generalization performance.

TABLE IV
TRAINING ALGORITHM COMPARISON (ONLINE MODE). ALL NUMBERS OF THE FORM $(m, s)$ ARE THE MEAN AND STANDARD DEVIATION VALUES

| Algorithm | Iterations | Training Error | Converged |
|---|---|---|---|
| bp | (9.85e+02, 4.20e+01) | (1.77e-03, 2.02e-03) | 4 |
| cprs | (8.96e+02, 1.91e+02) | (4.14e-04, 4.68e-04) | 11 |
| sed | (1.00e+03, 0.00e+00) | (7.62e-04, 5.74e-04) | 0 |
| swnp | (9.40e+02, 1.37e+02) | (5.50e-04, 4.63e-04) | 5 |
| wp | (7.80e+02, 2.52e+02) | (1.74e-04, 1.95e-04) | 12 |

TABLE V
TESTING COMPARISON (ONLINE MODE). ALL NUMBERS OF THE FORM $(m, s)$ ARE THE MEAN AND STANDARD DEVIATION VALUES

| Algorithm | Testing Error | Correct (%) |
|---|---|---|
| bp | (1.50e-03, 2.20e-03) | (85, 17) |
| cprs | (5.01e-04, 5.72e-04) | (92, 11) |
| sed | (5.68e-04, 5.24e-04) | (91, 8) |
| swnp | (5.53e-04, 4.47e-04) | (93, 7) |
| wp | (4.10e-04, 1.79e-04) | (97, 3) |

### A. Combined Search Algorithm

The CSA [11] employs two minimization strategies, namely modified weight perturbation and random search. Modified weight perturbation is a local search and the random search algorithm is a nonlocal search technique. CSA can be described by the following pseudocode.

```
while not converged
{
    /* modified weight perturbation */
    for i = 1 to 10
    {
        for each weight w
        {
          wsave = w;
          w = w + DELTA;
          /* DELTA is usually set to 1 */
          evaluate error;
          if error has not improved
            w = wsave;
        }
    }
    /* random search algorithm */
    for i = 1 to 30
    {
        for each weight w
        {
          wsave = w;
          w = uniformly distributed
          random number;
          evaluate error;
          if error has not improved
            w = wsave;
        }
    }
}
```

The CSA algorithm is very simple and the results obtained are surprisingly good, with convergence being very fast for small problems. Although CSA has been successfully used to

TABLE VI
RESULTS OF APPLYING KAKADU TO THE XOR PROBLEM (6.9 $\mu$W)

| Input (Volts) | | | Desired Output | Output (Volts) |
|---|---|---|---|---|
| 0.2 | 0.0 | 0.0 | 0.0 | 0.031 |
| 0.2 | 0.2 | 0.0 | 0.2 | 0.215 |
| 0.2 | 0.0 | 0.2 | 0.2 | 0.173 |
| 0.2 | 0.2 | 0.2 | 0.0 | 0.032 |

TABLE VII
RESULTS OF APPLYING KAKADU TO THE PARITY THREE-BIT PROBLEM (9.0 $\mu$W)

| Input (Volts) | | | | Desired Output | Output (Volts) |
|---|---|---|---|---|---|
| 0.2 | -0.1 | -0.1 | -0.1 | -0.1 | -0.103 |
| 0.2 | -0.1 | -0.1 | +0.1 | +0.1 | +0.952 |
| 0.2 | -0.1 | +0.1 | -0.1 | +0.1 | +0.103 |
| 0.2 | -0.1 | +0.1 | +0.1 | -0.1 | -0.098 |
| 0.2 | +0.1 | -0.1 | -0.1 | +0.1 | +0.103 |
| 0.2 | +0.1 | -0.1 | +0.1 | -0.1 | -0.073 |
| 0.2 | +0.1 | +0.1 | -0.1 | -0.1 | -0.090 |
| 0.2 | +0.1 | +0.1 | +0.1 | +0.1 | +0.105 |

train Kakadu, it is expected that performance would degrade rapidly for larger neural networks.

### B. XOR

XOR has been a benchmark problem for neural networks because it is a simple yet highly nonlinear application. The minimum network size which can solve this problem is (3, 2, 1) with one input being a bias. To make Kakadu behave like a smaller network, the weight values for the unconnected synapses are set to zero. Kakadu was successfully trained on this problem, results of this test being shown in Table VI.

The power consumption for these neural-network training problems was measured after the outputs had settled to the final output value. This is the static consumption and includes the chip plus the off-chip neuron dissipation. For XOR, this figure was 6.9 $\mu$W at 3 V and the standard 6.63 nA. The same problem has been trained with bias currents down to 3.5 nA.

The settling time from a change in the inputs until the output reaches 90% of the final value (the Kakadu chip driving a 2 pF active probe) was typically 30 $\mu$s. Of course, a higher load capacitance will increase the settling time of the chip.

### C. Parity (Three-Bit)

Another nonlinear benchmark test is the three-bit parity problem which can be thought of as XOR in three dimensions. In this example, a bipolar coding was used instead of the unipolar coding of the XOR problem to show that Kakadu is capable of both.

Kakadu was successfully trained using a (4, 3, 1) network, the results of the experiment being shown in Table VII. The quiescent power consumption for this problem was 9.0 $\mu$W.

### D. Parity (Four-Bit)

Another nonlinear benchmark test is the four-bit parity problem which can be thought of as XOR in four dimensions. This test was successfully trained using a (5, 4, 1) network,

TABLE VIII
RESULTS OF APPLYING KAKADU TO THE PARITY FOUR-BIT PROBLEM (15.6 $\mu$W)

| Input (Volts) | | | | | Desired Output | Output (Volts) |
|---|---|---|---|---|---|---|
| 0.2 | -0.1 | -0.1 | -0.1 | -0.1 | -0.1 | -0.071 |
| 0.2 | -0.1 | -0.1 | -0.1 | +0.1 | +0.1 | +0.071 |
| 0.2 | -0.1 | -0.1 | +0.1 | -0.1 | +0.1 | +0.066 |
| 0.2 | -0.1 | -0.1 | +0.1 | +0.1 | -0.1 | -0.085 |
| 0.2 | -0.1 | +0.1 | -0.1 | -0.1 | +0.1 | +0.061 |
| 0.2 | -0.1 | +0.1 | -0.1 | +0.1 | -0.1 | -0.073 |
| 0.2 | -0.1 | +0.1 | +0.1 | -0.1 | -0.1 | -0.059 |
| 0.2 | -0.1 | +0.1 | +0.1 | +0.1 | +0.1 | +0.071 |
| 0.2 | +0.1 | -0.1 | -0.1 | -0.1 | +0.1 | +0.141 |
| 0.2 | +0.1 | -0.1 | -0.1 | +0.1 | -0.1 | -0.117 |
| 0.2 | +0.1 | -0.1 | +0.1 | -0.1 | -0.1 | -0.103 |
| 0.2 | +0.1 | -0.1 | +0.1 | +0.1 | +0.1 | +0.051 |
| 0.2 | +0.1 | +0.1 | -0.1 | -0.1 | -0.1 | -0.056 |
| 0.2 | +0.1 | +0.1 | -0.1 | +0.1 | +0.1 | +0.022 |
| 0.2 | +0.1 | +0.1 | +0.1 | -0.1 | +0.1 | +0.085 |
| 0.2 | +0.1 | +0.1 | +0.1 | +0.1 | -0.1 | -0.024 |

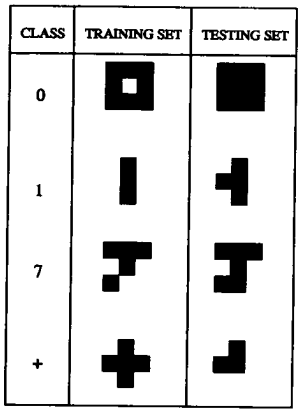



Fig. 9. Character recognition training and testing sets. Kakadu was trained on the training set and then tested with the corrupted characters in the right-hand column.

the results of the experiment being shown in Table VIII. The quiescent power consumption for this problem was 15.6 $\mu$W.

In this example, a satisfactory result could not be achieved by the direct optimization of (10) on the test chip. A mathematical model of the chip was derived from (3), and weights were allowed to be floating point values within the maximum range of the synapse values (i.e., $-31 \leq w \leq 31$). This enabled training to be performed in the absence of quantization effects and makes (10) continuous in this range. When the chip was trained using the starting values thus obtained, it converged quickly to a solution.

### E. Pattern Recognition

Kakadu was tested on a simple character recognition problem (Fig. 9). A (10, 6, 4) network was divided into a bias unit and a 3 x 3 pixel array. The network was trained (bias current 4.4 nA) on the characters "0," "1," "7," and "+," each output being assigned to one character. After training, a single bit in each character was corrupted and the network output passed through a "winner take all" decision to determine the network's classification of the corrupted character. The results of this experiment (shown in Table IX and Fig. 9) show that

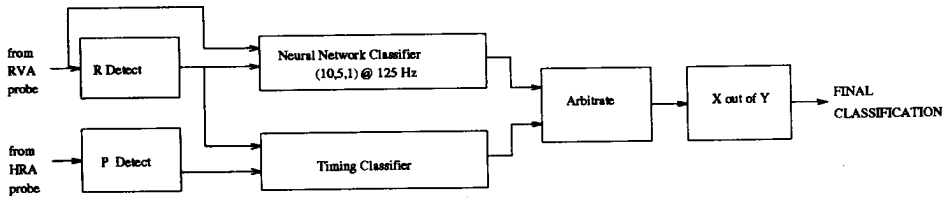



Fig. 10. Overview of the MATIC classification system. The *QRS* and $P$ wave detectors detect ventricular and a trial depolarizations, respectively ($R$ and $P$ waves).

TABLE IX
RESULTS OF APPLYING KAKADU TO THE CHARACTER RECOGNITION PROBLEM (22.5 $\mu$W). IN THE "INPUT" COLUMN A "1" DENOTES AN INPUT VOLTAGE OF 0.1 V AND A BLANK IS −0.1 V. AN ADDITIONAL BIAS INPUT OF 0.2 V WAS USED

| *Input* | | | *Output* | | | | *Class* |
|---|---|---|---|---|---|---|---|
| 1 1 1 | 1 1 1 | 1 1 1 | -0.12 | -0.11 | -0.09 | +0.10 | 0 |
| 1 1 1 | 1 1 1 | 1 1 1 | -0.10 | -0.04 | -0.16 | +0.09 | 0 |
| | 1 1 1 | | -0.10 | -0.10 | +0.10 | -0.10 | 1 |
| 1 | 1 1 1 | | -0.05 | -0.09 | +0.00 | -0.13 | 1 |
| 1 1 | 1 1 | 1 | -0.10 | +0.09 | -0.10 | -0.10 | 7 |
| 1 1 | 1 1 | 1 | -0.16 | +0.03 | -0.10 | -0.02 | 7 |
| 1 | 1 1 1 | 1 | +0.10 | -0.10 | -0.09 | -0.09 | + |
| 1 | 1 1 | | +0.06 | -0.08 | -0.02 | -0.15 | + |

Kakadu was able to correctly classify patterns that it had not been trained on. Kakadu draws 22.5 $\mu$W during this test.

## V. MATIC EXPERIMENT

### A. MATIC Algorithm

The MATIC algorithm [2] classifies arrhythmias based on timing and morphological features (see Fig. 10). Inputs to the classifier are voltage levels obtained from temporary catheters implanted on the surface of the heart, one in the high right atrium (HRA) and one in the right ventricular apex ventricle (RVA). Such signals are known as intracardiac electrograms (ICEG). The raw signals are then bandpass filtered (0.5–100 Hz), and this becomes the input to the MATIC algorithm. MATIC can identify four different types of arrhythmia, namely ventricular fibrillation (VF), ventricular tachycardia (VT), supraventricular tachycardia (SVT), and normal sinus rhythm (NSR). These four different arrhythmias correspond to the four different therapies available in an ICD, which are to apply a high energy shock (defibrillate), pace the ventricles, pace the atrium, or do nothing.

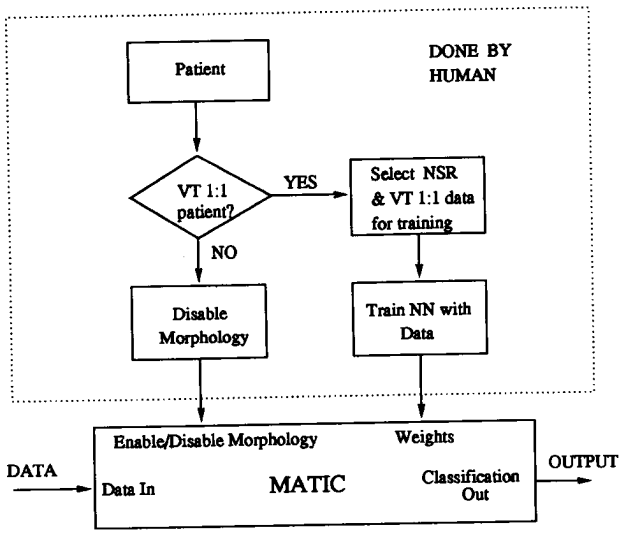



Fig. 11. Flow chart/signal diagram of the MATIC configuration procedure. In the case of VT 1:1 patients training data must be selected and the neural network trained to produce a set of weights. Note that morphology is only enabled for VT 1:1 patients.

The $R$ and $P$ wave detectors are peak detectors which are used to identify depolarizations in the RVA and HRA channels, respectively. From the output of the $R$ and $P$ wave detectors, the timing classifier can determine the depolarization sequences of the heart. From this timing information, reliable classification can be made for most arrhythmias. A certain type of VT, called ventricular tachycardia with 1:1 retrograde conduction (VT 1:1) cannot be classified based on timing between two channels. VT 1:1, however, is often characterized by a change in morphology of the RVA signal and a neural network is used to recognize normal, and VT 1:1 signals for a particular patient. The timing and morphology classifiers run in parallel, and the results are combined using simple arbitration logic to produce a final classification.

*MATIC Configuration:* Most patients do not require morphology classification, as timing is sufficient to identify the arrhythmia. For some patients, however, morphological classification is required. A human must configure the MATIC system before it is used to identify whether morphology is required. MATIC is used with morphology only for VT 1:1 patients. In the case of non-VT 1:1 patients, the arbitration logic discards all results from the neural network. A flow chart/block diagram of this configuration process is shown in Fig. 11.

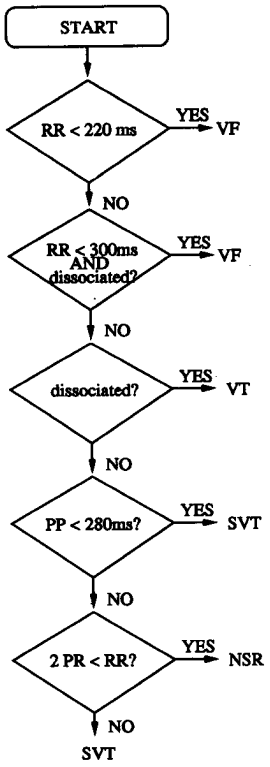



Fig. 12. Flowchart of timing based decision logic (PP = time interval between the previous two $P$ waves, RR = interval between previous $R$ waves, PR = time from last $P$ wave to $R$ wave). Note that "dissociated" means PP/RR > 1.5.

Configuration involves deciding if the patient is a VT 1:1 patient or not and if so, four samples each of the patient's NSR and VT 1:1 morphology must be provided to serve as templates for the morphology classifier. After training, the weights need not be changed.

After configuration, the MATIC system is fully automatic, taking the filtered data as input and producing NSR, SVT, VT, and VF classifications as output.

*Timing Logic:* The timing logic used in Kakadu can be described by the flowchart shown in Fig. 12. The timing logic first computes the interval between $R$ waves (RR interval), the interval between $P$ waves (PP interval) and the interval between the last $P$ wave and the last $R$ wave (PR interval). From these three numbers, the timing logic can identify four different types of arrhythmia, namely VF, VT, SVT, and NSR. These four different arrhythmias correspond to the four different therapies available in an ICD, which are to apply a high energy shock (defibrillate), pace the ventricles, pace the atrium, or do nothing. It is a very simple classifier, implemented using a decision tree which classifies the sequences much as a human would.

*Neural Network Morphology Classifier:* A typical VT 1:1 patient is shown in Fig. 13. It can be easily seen in the figure that the first three peaks (QRS complexes) are a different morphology to the last five QRS complexes. These patterns correspond to NSR and VT, respectively, and the neural network is used to recognize these rhythms.

A window of samples is formed via a delay line as illustrated in Fig. 14. The neural network can be thought of as running

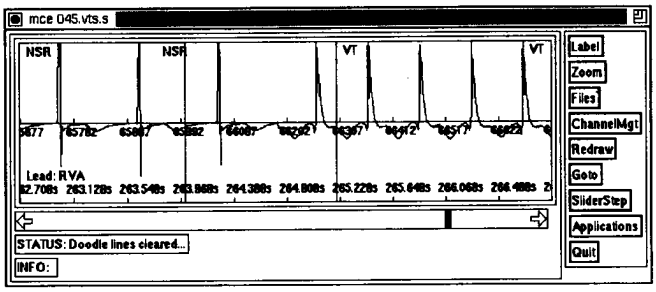



Fig. 13. VT with 1:1 retrograde conduction. Note how the morphology of the signal changes between normal rhythm (on the left) and VT 1:1 (on the right).

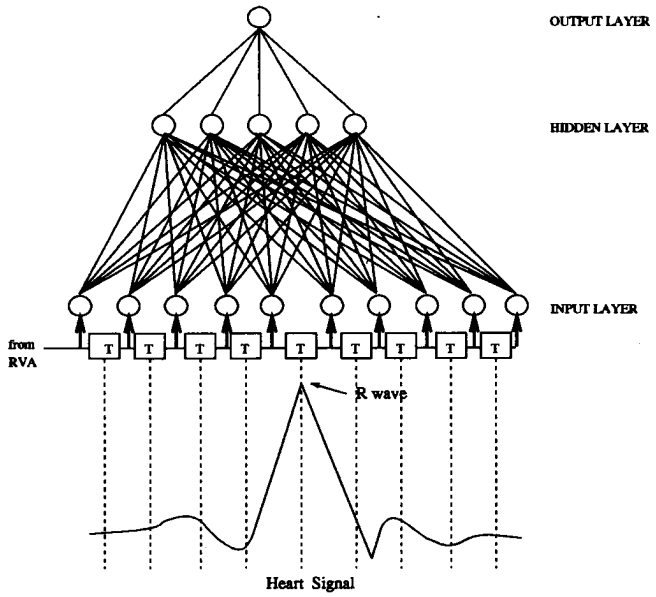



Fig. 14. Block diagram of the neural network used to perform morphology classification. The heart signal is centered within the neural network through synchronization with the $R$ wave.

continuously, and the output of the network is read when the QRS complex is centered in the 10 sample window formed by the delay line. The $R$ wave detector is used to detect the middle of a QRS complex and hence center the QRS complex.

To construct a training set for a given patient, four normal and four VT 1:1 rhythms are hand selected. Normal rhythms were trained to have an output of 0.0 V and VT 1:1 rhythms 1.0 V. The weights used by the MATIC morphology classifier are then obtained by training the neural network on this data set. When ICEG signals are applied to the neural network, a VT 1:1 morphology is assumed present if the output becomes greater than 0.8 V.

*Post Processing:* The post processor takes the output of both the timing and morphology algorithms and assigns an intermediate output class given by the function

$$\text{CLASS} = \begin{cases} \text{VT} & \text{if morphology} > 0.0 \text{ and timing} \neq \text{VF} \\ \text{timing} & \text{otherwise.} \end{cases}$$

The output is then passed through an $X$ out of $Y$ detector which outputs a classification only if five out of the last six output classes from the post processor are the same.

TABLE X
SUBCLASS CONFUSION MATRIX OF THE MATIC CLASSIFICATION USING THE KAKADU CHIP FOR MORPHOLOGY. NOTE THAT 99.6% OF CLASSIFICATIONS ARE CORRECT. THE ROWS REPRESENT THE HUMAN CLASSIFIED SUBCLASS AND THE COLUMNS REPRESENT THE MATIC CLASSIFICATION. HUMAN SUBCLASS CLASSIFICATIONS ARE COLLAPSED INTO SUPERCLASSES IN THE ROWS ACCORDING TO THE "CLASS" COLUMN. THE "REGIONS" COLUMN IS THE TOTAL NUMBER OF STABLE RHYTHMS FOR THAT SUBCLASS

| Subclass | Class | Regions | NSR | SVT | VT | VF |
|---|---|---|---|---|---|---|
| NSR | NSR | 64 | 5605 | 4 | 2 | 0 |
| ST | NSR | 10 | 1535 | 24 | 2 | 1 |
| SVT | SVT | 9 | 0 | 1022 | 0 | 0 |
| AT | SVT | 2 | 0 | 52 | 0 | 0 |
| AF | SVT | 3 | 0 | 165 | 0 | 0 |
| VT | VT | 9 | 0 | 0 | 332 | 0 |
| VT 1:1 | VT | 10 | 2 | 0 | 1253 | 0 |
| VF | VF | 6 | 0 | 0 | 2 | 196 |
| VTF | VF | 9 | 0 | 2 | 0 | 116 |

TABLE XI
SUBCLASS CONFUSION MATRIX OF THE MATIC CLASSIFICATION USING KAKADU. 98.4% ARE CORRECTLY CLASSIFIED

| Class | Class | NSR | SVT | VT | VF |
|---|---|---|---|---|---|
| NSR | NSR | 1114 | 0 | 18 | 0 |
| VT 1:1 | VT | 0 | 16 | 974 | 1 |

### B. MATIC Results

The MATIC system was used on a database of 12483 QRS complexes recorded from 67 patients [12] during electrophysiological studies (EPS) where temporary catheters were placed under fluoroscopic guidance in the RVA and the HRA. These signals were sampled at 1000 Hz and recorded.

The MATIC algorithm was applied with morphology algorithm being implemented on the Kakadu chip (via Jiggle) and the rest of the algorithm implemented in software. Note that to facilitate debugging, the delay line of Fig. 14 was implemented in software. An experimental prototype of a bucket brigade device delay line was implemented on an earlier chip [13] and found to have a charge transfer inefficiency of 0.035% which would be more than adequate for this application.

The database was "played" through this hybrid hardware/software system to obtain the results which are tabulated in Table X. The results of classifying only the VT 1:1 patients (the neural network is not used for other patients) is shown in Table XI.

To compare the results of the experiment with that of a digital neural network, a standard two-layer perceptron (TLP) of the same network size was first trained. The results of this are shown in Table XII. The TLP network has marginally better performance than the Kakadu network, and this is mostly due to the limited precision weight range available on the Kakadu chip (six bits).

The power consumption of Kakadu for the 10 VT 1:1 patients are shown in Table XIII. The maximum power consumption of the chip was 25 $\mu$W for the patients studied. The propagation delay of the Kakadu chip is approximately 30 $\mu$s, and Kakadu has negligible (<100 pA) power consumption at zero bias. If a conservative value of 1000 $\mu$s for propagation is allowed, the energy consumed is 25 nJ per feedforward operation. The bias to the chip can be turned off when it is not being used (99.9% of the time), and so the average power consumption of the system (assuming the normal heart rate is 1 Hz) can be reduced by a factor of 1000 to less than 25 nW.

TABLE XII
CONFUSION MATRIX OF MATIC CLASSIFIER USING A NORMAL THREE-LAYER PERCEPTRON IMPLEMENTED USING DOUBLE PRECISION FLOATING POINT ARITHMETIC. NOTE THAT 99.2% OF CLASSIFICATIONS ARE CORRECT. THE ROWS REPRESENT THE HUMAN CLASSIFIED CLASS AND THE COLUMNS REPRESENT THE MATIC CLASSIFICATION

| Class | Class | NSR | SVT | VT | VF |
|---|---|---|---|---|---|
| NSR | NSR | 1319 | 0 | 0 | 0 |
| VT 1:1 | VT | 1 | 15 | 1017 | 4 |

TABLE XIII
POWER CONSUMPTION OF KADADU CHIP FOR THE 10 VT 1:1 PATIENTS

| Patient | Power ($\mu W$) |
|---|---|
| 1 | 18.3 |
| 2 | 16.5 |
| 3 | 12.6 |
| 4 | 13.8 |
| 5 | 20.1 |
| 6 | 21.6 |
| 7 | 9.0 |
| 8 | 15.6 |
| 9 | 13.5 |
| 10 | 24.0 |

## VI. DISCUSSION

The problem of arrhythmia classification by morphology is not a particularly difficult problem if one is given a reasonable power budget. An ICD, however, requires very low power consumption, and this precludes the use of most arrhythmia classification algorithms.

Neural networks have been successfully applied to pattern recognition problems in many areas of signal processing. Their regular and parallel architecture make them suitable for VLSI implementation and they can be implemented efficiently using analog techniques.

Neural networks are particularly suitable for use in implantable devices since analog computation leads to small area, and by operating transistors in the subthreshold region, low power consumption is achieved. These design ideas were used to implement the MATIC system which has the advantages of neural network classifiers, while maintaining low power consumption.

## VII. CONCLUSION

We have described the MATIC system which classifies arrhythmias based on a hybrid decision tree and neural network approach. Because a neural network is computationally expensive, a low-power analog VLSI neural-network chip was designed and successfully trained to perform this task. The system can classify a large database of arrhythmias to an accuracy of 98.4% while consuming an average power of less than 25 nW.

## ACKNOWLEDGMENT

The authors would like to thank the reviewers for their constructive comments.

## REFERENCES

[1] T. Fahraeus, C. Lassvik, and C. Sonnhag, "Tachycardias initiated by automatic antitachycardia pacemakers," *PACE*, pp. 1049–1054, Nov. 1984.

[2] P. H. W. Leong and M. A. Jabri, "Matic—An intracardiac tachycardia classification system," *PACE*, Sept. 1992.

[3] R. Lippmann, "An introduction to computing with neural nets," *IEEE Acoust., Speech, Signal Process. Mag.*, pp. 4–22, Apr. 1987.

[4] C. A. A. Bastiaansen, J. Wouter, D. Groeneveld, H. J. Schouwenaars, and H. A. H. Termeer, "A 10-b 40 mhz 0.8 $\mu$m CMOS current-output D/A converter," *IEEE J. Solid-State Circuits*, vol. 26, no. 7, pp. 917–921, July 1991.

[5] P. H. W. Leong and J. G. Vasimalla, Sydney University Electrical Engineering, JIGGLE User's Manual, Sedal Tech. Rep. PL-0892, 1992.

[6] J. Alspector, R. Meir, B. Yuhas, A. Jayakumar, and D. Lippe, "A parallel gradient descent method for learning in analog VLSI neural networks," in *Advances in Neural Information Processing Systems (NIPS) 5*. San Mateo, CA: Morgan Kaufmann, 1993, pp. 836–844.

[7] G. Cauwenberghs, "A fast stochastic error-descent algorithm for supervised learning and optimization," in *Advances in Neural Information Processing Systems (NIPS) 5*. San Mateo, CA: Morgan Kaufmann, 1993, pp. 244–251

[8] B. Flower and M. A. Jabri, "Summed weight neuron perturbation: An $o(n)$ improvement over weight perturbation," in *Advances in Neural Information Processing Systems (NIPS) 5*. San Mateo, CA: Morgan Kaufmann, 1993, pp. 212–219.

[9] M. A. Jabri and B. Flower, "Weight perturbation: An optimal architecture and learning technique for analog VLSI feedforward and recurrent multilayer networks," *Neural Computa.*, vol. 3, no. 4, pp. 546–565, Dec. 1991.

[10] M. A. Jabri, "Practical performance of learning algorithms for analog multilayer perceptrons," SEDAL Tech. Rep., Oct. 1993.

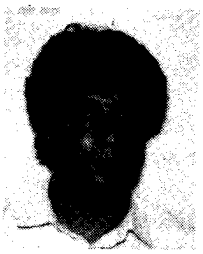
[11] Y. Xie and M. A. Jabri, "Training limited precision feedforward neural networks," in *Proc. 3rd Australian Conf. Neural Networks*, 1992, pp. 68–71.

[12] P. H. W. Leong, "Arrhythmia classification using low-power VLSI," Ph.D. dissertation, University of Sydney, Dec. 1992.

[13] P. H. W. Leong and M. A. Jabri, "A VLSI neural network for arrhythmia classification," in *Proc. Int. Joint Conf. Neural Networks*, vol. II, Baltimore, MD, 1992, pp. 678–683.

**Philip H. W. Leong** (S'88–M'88) received the B.Sc., B.E., and Ph.D. degrees from the University of Sydney, Australia, in 1986, 1988, and 1993, respectively.

In 1989, he was a Research Engineer at AWA Research Laboratory, Sydney. From 1990 to 1993, he was a postgraduate student and Research Assistant at the Systems Engineering and Design Automation Laboratory (SEDAL), Sydney University Electrical Engineering, where he worked on low-power analog VLSI circuits for arrhythmia classification. In 1993, he was a Consultant to SGS Thomson Microelectronics in Milan, Italy. He joined the Department of Electrical Engineering, University of Sydney, in 1994 where he is currently a Lecturer. He is the author of more than 20 technical papers and one patent. His research interests include analog integrated circuits and signal processing.



**Marwan A. Jabri** (S'84–M'88–SM'94) received the License de Physique and a Maitrise de Physique from Universite de Paris VII, France in 1981 and 1983, respectively and the Ph.D. degree in electrical engineering in 1988 from Sydney University, Australia.

He is a Reader at the University of Sydney, Department of Electrical Engineering, and Director of its Systems Engineering and Design Automation Laboratory (SEDAL). He was a Visiting Scientist at AT&T Bell Laboratories in 1993. He is author, co-author, and editor of three books, over 100 papers and four patents. His research interests include digital and analog integrated circuits, neural computing, and machine intelligence.

Dr. Jabri is a recipient of the 1992 Australian Telecommunications and Electronics Research Board (ATERB) Outstanding Young Investigator medal. He is Vice-president of the Asia Pacific Neural Network Assembly.