Adaptive logic circuits

By I. Aleksander and R. C. Albrow*

The paper gives a survey of physically realisable adaptive circuits. Past work on biological models and their descendents is reviewed, these being largely analogue. A purely digital adaptive model or Adaptive Logic Circuit (ALC) is introduced. Six examples of design techniques are given in order to illustrate the way in which the processing properties of ALC networks may be determined. These include the description of an adaptive microcircuit which has been developed as a building brick for ALC systems. Comparisons with the analogue devices are made and the topic of reliability is touched on.

This paper was presented at the British Computer Society First National Symposium on Logic Design, held at Reading University in July, 1967.

We are generally familiar with logic systems containing AND, OR, NOT, NAND, etc., gates. Due to microcircuit technology we are now becoming accustomed to modules that contain these gates and, in some cases, have predetermined logic duties such as binary addition, storage, pulse generation and counting. We are also beginning to see 'multipurpose' devices where, say, a shift register may be turned into a binary counter by the application of an appropriate bias. Adaptive Logic Circuits (ALCs) go one step further in this direction. They are designed to perform a very large number of logic functions of which they adopt one at any one time. Therefore, these turn out to be not only very versatile logic circuits but also building blocks for a range of self-organizing tasks in which the logic system *adapts* to some desired function *in actual use.*

Such systems have been studied in the past under the guise of biological models where adaptivity appears as the property of 'learning'. This has resulted in electronic models where 'experience' is stored as the value of variable resistors and logical decisions are made by summing (or threshold) circuits. Such models are not very useful building bricks for adaptive systems mainly on account of the difficulty of manufacturing analogue stores.

This paper describes a purely digital adaptive model which uses binary storage and makes unrestricted logic decisions. Design examples of the model itself and networks containing it are discussed to indicate that ALCs can produce patterns of 'learning' behaviour similar, and possibly superior, to those of biological models.

We must define two salient points:

Definition 1: An *Adaptive Logic Circuit* consists of a set of binary inputs I and binary outputs F. The Boolean function relating F to I may be changed by a binary control signal and remain unchanged in the absence of such a signal.

Definition 2: *Adaptivity* is the property of absorbing information during a set of 'teaching' trials for improving the performance during subsequent use of the device.

* *Queen Mary College, Mile End Road, London, E.*1.

Biological models

In this section we concentrate on what appears to be the 'main stream' of biological modelling particularly with respect to the possible use of these models in the construction of data processing machinery. Figs. 1 to 4 are a guide for this purpose.

Fig. 1 shows a simplified diagram of the neuron: a nerve cell that is considered to be the primary element of living nervous systems. McCulloch and Pitts (1943) suggested that the simple model shown in **Fig. 2** accounts for most of the outward signal processing properties of the neuron. They argue that vast networks of these elements could conceivably perform data processing tasks akin to those of the adult brain. A human brain contains approximately 10^{10} neurons. The property that

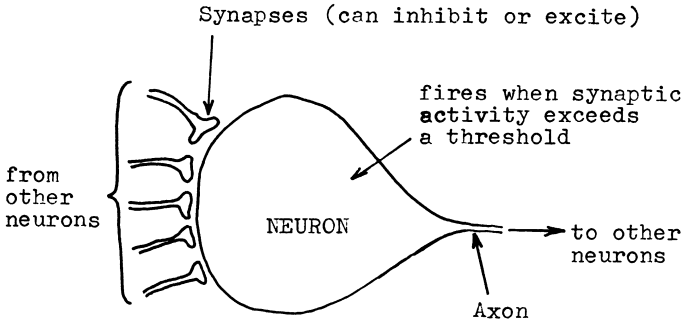

Fig. 1. A simplified neuron

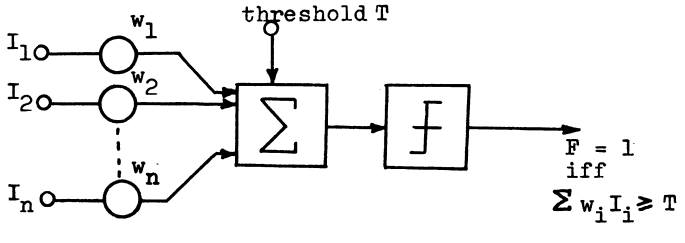

Fig. 2. The McCulloch and Pitts model

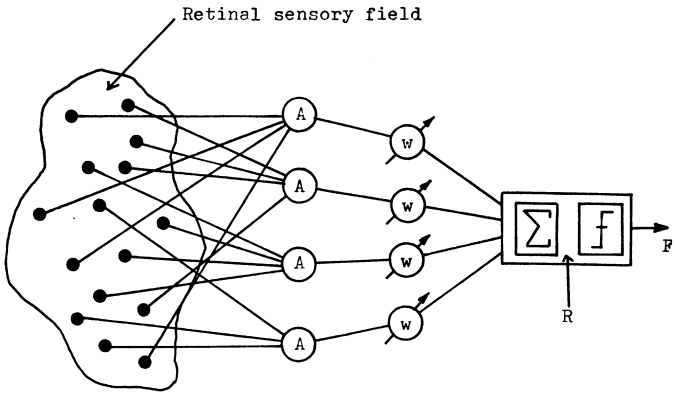

Fig. 3. The Perceptron

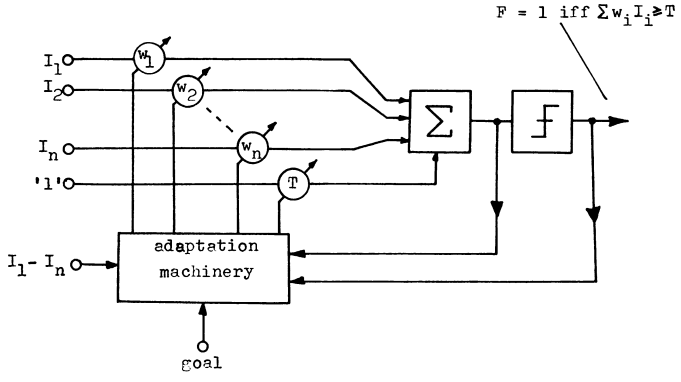

Fig. 4. The Adaline

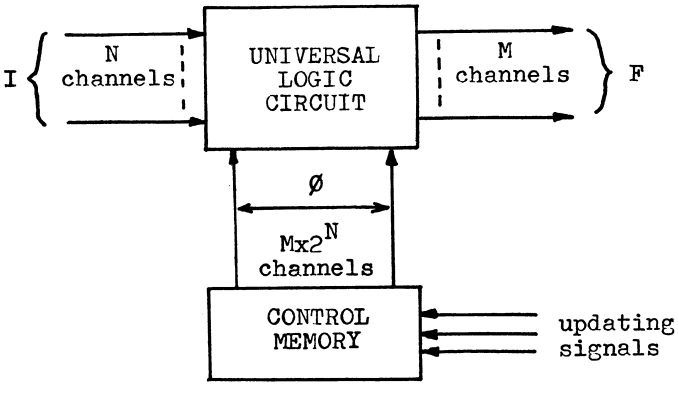

Fig. 5. Model for adaptive logic circuits

seems to be missing in this model is the mechanism whereby the interconnections in the net are formed to provide the desired function at the output terminals. Leaving aside the question of genetic evolution, some form of plasticity (viz. adaptivity) must be associated with the McCulloch and Pitts model to enable a net that is not performing the correct function (as, say, in a newly born infant) to *adapt* to the desired tasks (viz. it could 'learn').

Most credit for this kind of advance goes to Rosenblatt (1957) who presented a concept (rather than a model) called the *Perceptron* (see **Fig. 3**). A simple Perceptron consists of a sensory field, randomly connected to a set of threshold devices called *association units*, *A*. Groups of these are connected to *threshold response units*, *R*. Adaptation takes place by the *reinforcement of connections*. Physically, this requires variable resistor-like weights in the *connection paths*. It was shown by computer simulation and physical modelling that the adaptive behaviour of the Perceptron improves if the *A*-units are coupled to one another and if several layers of *A*-units are used.

Widrow (1961) studied a more practical model, the 'Adaline': a threshold element shown in **Fig. 4**. This again is a weight-dependent threshold circuit whose weights are adjusted by goal-seeking adaptation machinery. Widrow has used this model (simulated and physical) in tasks ranging from the balancing of an upright pendulum to playing the game of Blackjack (Widrow, 1966).

Other adaptive models have appeared in the literature, namely a Perceptron-like machine designed by Taylor (1959) at University College London, and two-input variable state devices such as the Neurotron, the Artron and the Reron (Lee, Pedelty, Snyder and Gilstrap, 1963) which employ statistically adjusted weights.

Our main objection to the above models is their dependence on analogue storage devices. The resistor weights of the Perceptron and the Adaline act as analogue stores of the function of the device. Similar stores are found in some form or other in the other models mentioned above. A study of analogue stores has been made by Nagy (1963), who considered magnetic, electromechanical and electrolytic components. A serious lack of useful devices is revealed, particularly of a kind which might be compatible with microcircuit techniques.

Much work has been done on adaptive systems that are implemented by digital computer simulation only. The literature on this subject is most extensive and well covered by a survey paper by Sklansky (1966). Of particular interest in this work is the Stella concept of Andreae and Gaines (1966), where the computer is used to simulate a physically realisable scheme. This has resulted in stochastic computing networks which use conventional microcircuit components and which have been shown to be useful in a type of adaptive logic system (Gaines, 1967).

The adaptive logic model

In view of the drawbacks encountered with analogue memories, our main aim is to design adaptive networks that employ binary storage and conventional decision logic. Furthermore we are not satisfied that the biological model (implemented as it might be with digital weights) is in any way optimal as an adaptive device. We thus resort to basic principles in defining our Adaptive Logic Circuit model shown in **Fig. 5**.

This contains two main components. The first is a universal logic circuit capable of *all* the logic functions between its sets of N input terminals I, and M output terminals F. Since this device can perform 2^{M2^N} logic

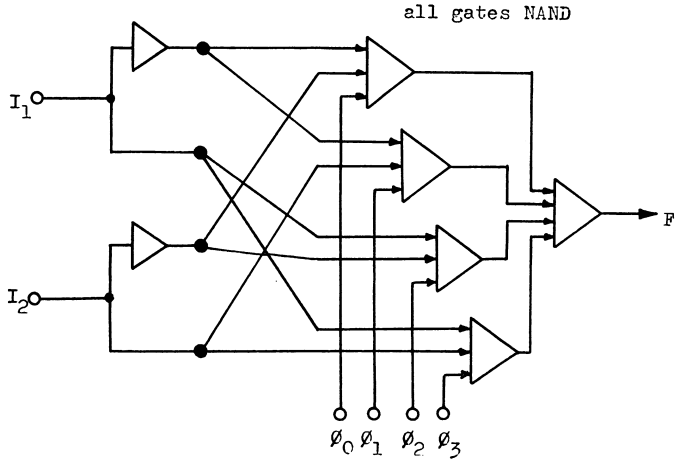

Fig. 6. Two-input logic circuit

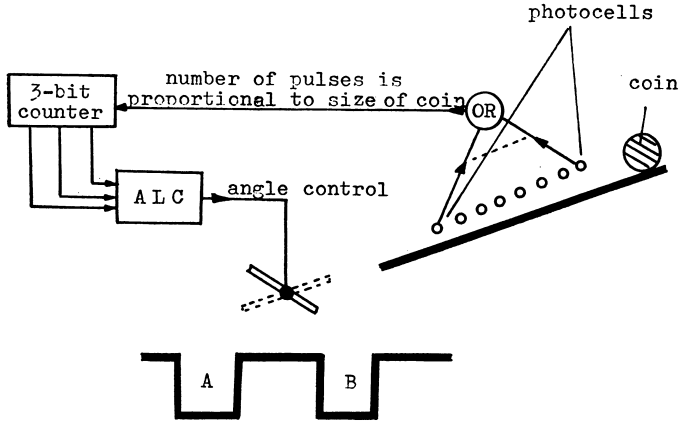

Fig. 7. Coin sorter

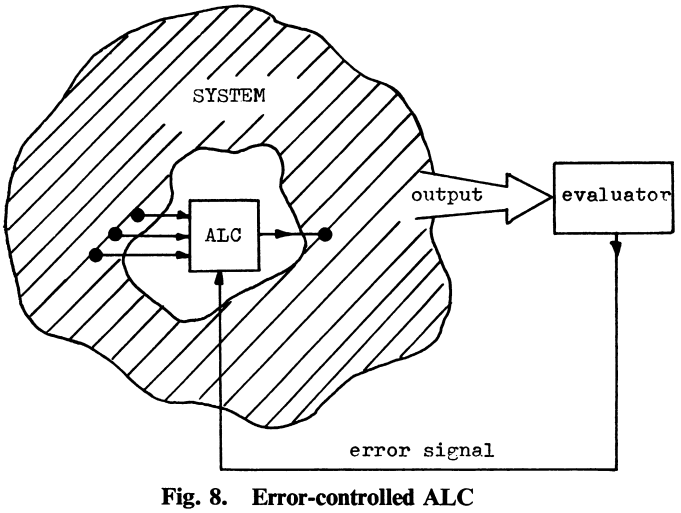

Fig. 8. Error-controlled ALC

functions, $M2^N$ bits of information must be supplied to it to indicate *which one* of these functions it should perform at any given time. This control information is provided in parallel on a set of $M2^N$ terminals ϕ. An example of a universal logic circuit in NAND form is shown in **Fig. 6** where $M = 1$ and $N = 2$. The general design of such circuits has been discussed elsewhere (Aleksander, 1966a, 1967).

The second part of the system is the control memory. This is a sequential circuit characterised by the following properties:

(i) It has at most $M2^N$ outputs.
(ii) It has one or more input channels for updating purposes.
(iii) It has 'essential memory' states. That is, it is capable of holding many output patterns for one particular input signal.

We note that:

This description includes ring counters, shift registers and standard flip-flop memories. A central part of our thesis is that *the design of any adaptive element may be translated into a sequential circuit design problem, the sequential circuit being the control memory of a universal logic circuit.*

This point is stressed in the rest of this paper which is concerned with six examples of adaptive element or network design.

Examples of adaptive logic circuit design

(a) *Function-searching ALCs*

The adaptive specification here is designed to cause the sequential memory to produce all the possible patterns at the control wires in response to an *error* signal at the input of the memory. On cancellation of the error signal the last found pattern at ϕ is held. A binary counter where the error signal is a train of pulses will act in this way. Similarly, a specific 'stoppable' autonomous sequential circuit which responds to a d.c. error signal may be designed (Aleksander, 1966b).

To discuss the adaptation characteristics of this and other examples we consider the following application. A machine (**Fig. 7**) is required for sorting eight different denominations of coins into two categories. There are 256 possible ways of doing this and our circuit must be able to adapt to any one of these. The denomination is sensed by a set of photocells in a chute arranged so that the number of pulses from the system is proportional to the size of the coin. From 1 to 8 pulses can be emitted, this number being stored in a 3-bit counter. The ALC must perform the two-way classification on the output of this counter and must therefore have $N = 3$ and $M = 1$. Assuming that 'training' trials consist of dropping coins down the chute and applying error signals until the right decision is made, it is evident that up to 256 training trials might have to be made in order to find the correct function. This is due to the fact that the system absorbs little information from the trials. The only advantage of this scheme is that the trainer *can* make errors since the sequential memory will come round repeatedly to the desired pattern if it should be missed during the training procedure (after another maximum of 256 trials).

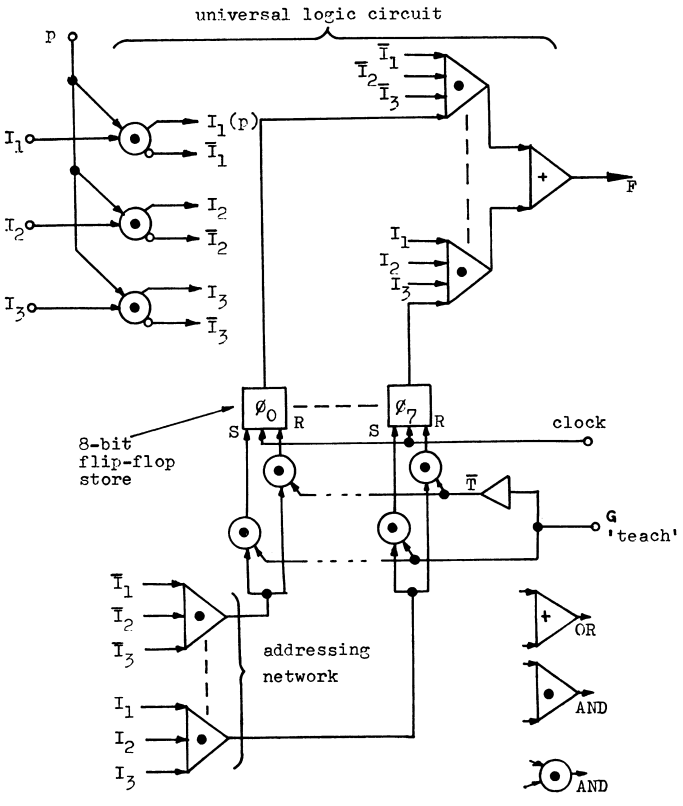

Fig. 9. Block diagram for SLAM 8

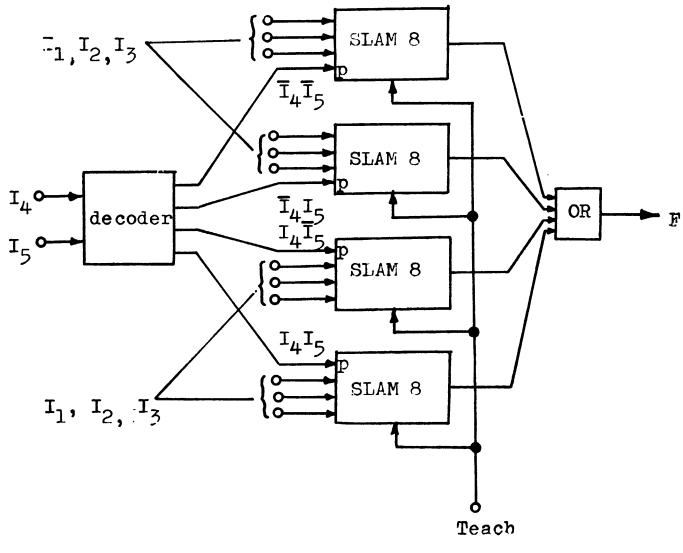

Fig. 10. SLAM 32

Another application of a function searching ALC is shown in **Fig. 8.** Here the ALC's performance is changed in order to improve the performance of the system as a whole. It is not necessary to measure the output of the ALC itself as is the case in some of the examples below.

(b) *'Lock-out' function-searching ALCs*

The search time in (a) may be shortened by locking out of future trials the bistable element that is responsible for the disappearance of the error in the present trial. In our example this would require teaching trials on only eight different coins since an error may be found only once for each coin. No training mistakes can be made and a method of resetting the bistables must be provided. It could be said that in this case that the sequential circuit absorbs information regarding *which* coin has been incorrectly classified, whereas in (a) it merely 'learns' whether its search is finished or not.

(c) *Input-addressed ALCs: Slam-8*

The same effect as in (b) may be achieved by addressing the bits of the memory from the input set I as shown in **Fig. 9.** Here the *desired* input (say G where G may be 0 or 1) is fed to a 'teach' terminal and used to set or reset the *required* control wire. For example, an input pattern at I (say $N = 3$) will not only excite three out of four inputs to a minterm gate in the ULC (see Fig. 9) but also address the flip-flop whose output forms the fourth input to the same minterm gate. This flip-flop is set to 0 or 1 depending on whether a 0 or a 1 is required at the output. Should we require to drive such a device from an error signal E we note that $G = \bar{E}F + E\bar{F}$ where F is the actual ALC output.

A 3-variable device of this kind has been made in microcircuit form. This device has been called Slam-8: a *S*tored *L*ogic *A*daptive *M*icrocircuit with 8 bits of storage. This is an all-MOST device containing 180 MOSTs in a TO5, 14-lead encapsulation (Albrow, Aleksander and Noble, 1967). Slam-8s may be interconnected to form ALCs for more than 3 variables. A 5-variable circuit is shown in **Fig. 10.** This is made possible by the gating terminal p (included in Slam 8) which must be energised to gate-in the inputs.

In the above we have reduced the maximum teaching time to q trials where, in general, $q = M2^N$. No form of generalisation is evident. Generalisation is the property of being able to classify correctly inputs not in the training set. In the next two examples we present two methods for reducing this time further and introducing some generalisation. Constrained networks of universal ALCs are used in the first case, whereas circuits with less than $M2^N$ bits of storage feature in the second. It should be pointed out that the details and properties of these constraints are the subject of present research, thus only the principles involved are illustrated.

(d) *Network-constrained ALCs*

A constrained network has P inputs and R outputs and contains M, N-variable universal ALC's where

$$M2^N < R2^P.$$

This is essential in pattern recognition tasks where the required storage of $R2^P$ and consequent training time would be prohibitively large. An example of such a net

is a pattern classifier with a sensory field of R retinal points. The inputs of $\frac{R}{3}$ Slam-8's are connected in a random, but one-to-one manner to all the points of the sensory field. If the $\frac{R}{3}2^3$ $\left(\text{i.e. } \frac{8R}{3}\right)$ bits of storage in the Slam-8s are set up at random, it is expected that a pattern at the sensory field is likely to produce a random pattern of 0s and 1s at the Slam-8 outputs. Training to recognise patterns would then consist of feeding 1s to all the 'teach' terminals for one class of training patterns and 0s for the other. The de-randomised stores would tend to correlate with input patterns which are statistically similar to the training set, a measure of this correlation being the ratio of 0s to 1s at the output of the net.

This type of net is thought to be analogous to a simple Perceptron, and as a pure guess, if the outputs of the net were fed back to some of the inputs the analogy would extend to a cross-coupled Perceptron. A certain similarity to Probability State Variable devices is also noted (Lee *et al.*, 1963).

(e) *Storage-constrained ALCs*

Here we describe an ALC with a built-in classification algorithm. This is specified (almost arbitrarily) to make the ALC behave like Widrow's Adaline.

The built-in restriction in Adalines is that any classification is a *threshold* (or linearly separable) function of the inputs. For three variables there are 104 threshold functions in the total possible of 256. In systems with more variables the density of threshold functions in the total is even smaller. Such a device is able to generalize (i.e. take a good guess at a classification which is not part of the training set) *if* the input patterns are clustered together in the input space, that is, they have many variables in common. R. Mattson (1959) and others have shown that this threshold function splits the input space of the Adaline by means of a hyperplane.

Thomas (1967) has used feedback loops in the memory of a type (c) ALC to force *only* linearly separable functions on the device. These feedback loops are analogous to those which turn a 16-message binary memory into a 10-message decimal one. In the previous example they reduce a 256-message, 8-bit binary memory of the type (c) device to the 104-message memory required by an Adaline type of device. It is reasonable to assume that such an ALC will have the same generalization properties as Adaline.

Naturally, in digital circuits we are not limited to linear separators, and there is much scope for the design of ingenious generalisation algorithms.

(f) *Sequential ALCs*

So far, the behaviour of ALCs has been described as purely combinational (at least, as far as the input-output function is concerned). Clearly sequential functions could be considered in a similar way.

Firstly, by applying a sequence of 0s and 1s to the inputs of the ALCs, one input being delayed from the next by one clock time, we can make a universal digital filter (Huffman, 1956). This device could be made to adapt to a desired filtering action by the same error signal response as was used in the combinational case.

Secondly, one can envisage sequential networks where ALCs control the logic in the feedback paths and thus the internal states of the network. This is an exciting prospect for further work.

Adaline or ALC? A storage comparison

As was mentioned previously, the weights of an Adaline may be made up of binary elements. It is of interest to calculate the required storage capacity of such an arrangement and compare it to that of an ALC which performs linear separations (this is the device discussed in (e) above, to which we now apply the mnemonic LSALC: Linearly Separable ALC). Let us compare firstly an Adaline with 6 inputs to an LSALC with $N = 6$ and $M = 1$.

Muroga (1965) has shown that the maximum weight (in a set of integer weights) for a 6-input threshold device is 5. According to Widrow's training algorithm (3) a unit error must be adjustable by $\left(\frac{1}{N+1}\right)$ of a unit. Thus the weight of 5 must be divided into at least 35 steps in our case. The number of bits per weight in a 6-input digital-weight Adaline is the nearest integer above $\text{Log}_2 35$, namely, 6 in this case. Assuming that the threshold requires the same number of bits, the minimum total storage capacity of a 6-input Adaline is 42 bits.

Muroga (1965) and others have also calculated that the number of linearly separable functions of six variables is 15,028,134 which requires only 24 bits of storage in a LSALC device. This result can be generalised for large values of n (by a rather rough use of Muroga's results):

$$\frac{\text{No. of bits in a digital Adaline}}{\text{No. of bits in a LSALC}} = r$$

where r lies between 1 and 2.

This ratio indicates that redundant storage is used in Adaline mainly due to Widrow's training algorithm. It may well be that in an LSALC the storage might have to be greater than the minimum to include a built-in training algorithm.

The reliability of ALCs

ALCs are redundant in an interesting way. Taking Slam-8 as an example we can show that the device can perform a useful function even if some of its storage cells are damaged.

The device remains universal for two variables even if up to four of its storage cells are damaged. In some cases additional gates may be required depending on the nature of the damage. If the inputs to Slam-8 are i_1, i_2 and i_3 and the device is to be universal for two variables

A and B, **Table 1** shows the necessary transformation for three sets of four out of eight healthy stores each. Clearly, the device is universal in one variable even if up to six of the eight storage cells are damaged.

The question of ALC reliability may be looked at in a different, non-universal, sense. There is a finite probability of achieving an N-variable function even if some of the cells are damaged. For example, assuming that the truth table of a function contains X 0s, Y 1s and D 'don't cares', the probability of achieving this function when one cell is damaged is

$$\left(\frac{X+D}{X+Y+D}\right)P+\left(\frac{Y+D}{X+Y+D}\right)(1-P)$$

where P is probability of the damaged cell always producing a 0 at its control wire. Thus the probability that a Slam-8 with one damaged cell ($P=\frac{1}{2}$, say) can act as a three-input full adder is 50% since $X=Y=4$ and $D=0$.

Table 1

	HEALTHY STORES i_1	i_2	i_3	TRANSFORMATION TO A, B	REMARKS
Set i	0	1	0	$i_1 = A$	i_3 is connected to 0; i_1 and i_2 being used as inputs.
	1	0	0	$i_2 = B$	
	1	1	0	$i_3 = 0$	
	1	0	0		
Set ii	1	1	1	$i_1 = i_3 = A$	i_1 and i_3 are joined to act as one input; i_2 being the other.
	1	0	1	$i_2 = B$	
	0	1	0		
	0	0	0		
Set iii	0	0	0	$i_1 = A$	i_1 and i_2 are used as input; an AND gate being required for i_3.
	0	1	0	$i_2 = B$	
	1	1	1	$i_3 = AB$	
	1	0	0		

Conclusions

Here we list those aspects of ALCs which have been investigated to date and follow this up with a list of work which remains to be done.

* We are satisfied that our universal logic circuit plus the sequential memory is a model into which we can translate desired adaptive actions and which is functionally equivalent to less realisable schemes.
* The nature of the *universal logic circuit* part of the model is well understood and does not require much further work.
* The design of universal *function searching* sequential memories is understood and can easily be realised. Their main use is in adaptive tasks under the action of an error signal, either with a dubious teacher or with weakly specified systems (e.g. adaptation to unspecified tasks as in Fig. 8). The universal search is time-limited to, say, five input variables for 10 MH_z search speeds. These devices may also maintain the system performance in the face of varying system parameters.
* 'Lockout' and Slam-8 devices are less time-limited (in the order of about 30 input variables for the above search speeds).
* 'Lockout' devices may be 'trained' by an overall error signal but are suitable for 'once-and-for-all' learning only.
* Slam-8 devices require *more information regarding their own performance*, but may conceivably maintain the performance of a system in the face of varying system parameters.
* Redundancy in ALCs is a great asset with regard to their production yields and possible reliability in service.
* Non-adaptive equipment may utilise ALCs as versatile 'building bricks'.

As for the future the following immediate tasks appear.

* Optimal storage constraints must be found for *particular* tasks. These are likely to differ from task to task. Some attention should be given to constrained, function searching systems.
* The behaviour of non-universal networks using universal ALCs must be investigated and optimisation criteria found.
* The behaviour of ALCs in sequential networks remains to be investigated.
* Existing ALCs must be tested and compared in specific tasks (e.g. in learning controllers, pattern recognition machines, speech recognition, general purpose adaptive processors, etc.).

References

ALBROW, R. C., ALEKSANDER, I., and NOBLE, P. J. W. (1967). A universally adaptable monolithic logic module, *Electronic Communicator*, July/August 1967.

ALEKSANDER, I. (1966a). Design of universal logic circuits, *Electronics Letters*, Vol. 2, p. 319.

ALEKSANDER, I. (1967b). Self-adaptive universal logic circuits, *Electronics Letters*, Vol. 2, p. 321.

ALEKSANDER, I. (1967). Adaptive systems of logic networks and binary memories, *Proc. Spring Joint Comp. Conf.*, p. 707.

GAINES, B. R., and ANDREAE, J. H. (1966). A learning machine in the context of the general control problem, *Proc. 3rd IFAC Congress*, London.

GAINES, B. R. (1967). Stochastic computing, *Proc. Spring Joint Comp. Conf.*, p. 149.

HUFFMAN, D. A. (1956). The synthesis of linear sequential coding networks, *Third London Symposium on Information Theory.*

LEE, PEDELTY, SNYDER and GILSTRAP (1963). Theory of probability state variable systems, U.S. Govt. research report AD 427872.

MCCULLOCH, W. S., and PITTS, W. (1943). A logical calculus of the ideas imminent in nervous activity, *Bull. Math. Biophysics* Vol. 5, p. 115.

MATTSON, R. L. (1959). A self-organizing binary system, *Proc. Eastern Joint Comp. Conf.*, p. 212.
MUROGA, S. (1965). Lower bounds on the number of threshold functions and a maximum weight, *Trans. I.E.E.E.*, EC 14, p. 135.
NAGY, G. (1963). A survey of analog memory devices, *Trans. I.E.E.E.*, EC 12, p. 388.
ROSENBLATT, F. (1961). Principles of neurodynamics: Perceptrons and the theory of brain mechanisms, Spartan Books.
SKLANSKY, J. (1966). Learning systems for automatic control, *Trans. I.E.E.E.*, AC 11, p. 6.
TAYLOR, W. K. (1959). Pattern recognition by means of automatic analogue apparatus, *Proc. I.E.E.* (London), p. 198.
THOMAS, T. N. (1967). Proposals for a linearly separable adaptive logic circuit, Q.M.C. research note (available from the B.C.S. Logic Design Group repository).
WIDROW, B. (1966). 'Bootstrap learning' in threshold logic systems, *Proc. 3rd IFAC Congress*, London.
WIDROW, B., and HOFF, M. E. (1961). Adaptive switching circuits, *Wescon Convention Record, IRE.*

Book Review

Applied Regression Analysis, by N. R. DRAPER and H. SMITH, 1966; 407 pages. (New York: *John Wiley & Sons, Inc.*, 90s.)

It is probably true that to-day more than 90% of the time spent in applying statistical methods by means of computers is occupied by fitting to data relationships involving several variables, i.e. regression analysis. There is nothing surprising in this high proportion, for the technique of regression is attractive and simple as a descriptive technique in many important economic and industrial situations. Unfortunately, however, the interpretation of the results of the application of this technique can be a highly complex matter. Whilst admitting that the volume of activity in applying regression analysis has increased substantially since the introduction of computers, it is also necessary to realise that the amount of guidance provided for users of such a technique by textbooks has not increased commensurately. It is therefore refreshing to encounter a book which is not a conventional and theoretical introduction to regression analysis. As well as providing such guidance in good measure the book at present being reviewed has three particular virtues. These are an emphasis on the analysis of residuals, an account of current methods for selection of best regression models and a most welcome and uncommonly extensive section on non-linear regression. In those situations where multiple regression has been tried and found to give puzzling and unsatisfactory results, it has not been common practice in the past to evaluate the residuals or deviations from the model because of the computational exercise involved in such evaluation. The examination of residuals, it is shown, may be assisted by applying certain formalised procedures of the type given in this book. For this essential part of regression analysis graphical output is of course indispensable.

In an interesting chapter, various step-wise and stage-wise regression techniques are described with worked examples, and personal conclusions are drawn by the authors. The sheer quantity of computer output is staggering with some of the fifty pages spread rather thinly with small tables. To some extent, this amount of output is defensible on the grounds that in order to compare methods for selecting the best regression equation in a particular situation, a student usually must examine a considerable volume of output. It must be added in fairness to the authors that their discussion of selection techniques is very reasonable and they give sufficient warning of the dangers of applying these techniques in a routine manner.

Many non-statisticians will be interested in the survey of non-linear minimisation presented in the last chapter, and most of the recent papers appearing in this *Journal* and elsewhere are discussed here and cited in an excellent 'non-linear' bibliography.

The level of statistical knowledge assumed is not great and although matrices are used extensively to describe the procedures, they are carefully introduced and extended by illustration. Simple matrix properties are described and formulae are given for the inversion of special and small-order matrices. The book would be suitable for a course on the application of regression analysis assuming computing facilities were available. It is nevertheless doubtful whether this book could itself provide sufficient basis for training a specialist in this area, because of its lack of underlying theory. In its defence the book does usually refer to suitable texts and research papers, and the coverage of the bibliography is excellent. There are useful numerical examples for the student at the end of most chapters with solutions at the end of the book. Except for the arrangement of some of the computer output already mentioned, the presentation of the book is excellent. The book is also to be recommended as a general reference for anyone in the computing field who meets statistical applications, and for practical statisticians.

R. W. HIORNS (Oxford)