TECHNICAL REPORT NO. 315

Connectionist Logic Programming

by

Jonathan W. Mills

October 1990

COMPUTER SCIENCE DEPARTMENT
INDIANA UNIVERSITY
Bloomington, Indiana 47405-4101

Connectionist Logic Programming*

Jonathan Wayne Mills
Computer Science Department
Indiana University
Bloomington, Indiana 47405

Abstract

Logic programming has seen little application to connectionist research in AI. However, Kowalski's description of a logic program, algorithm = logic + control, can be extended into the connectionist domain by using hybrid digital-analog array processors that evaluate sentence schemata for multiple-valued propositional logics.

In connectionist logic programming, the "logic" part of Kowalski's description corresponds to a forest of sentence schemata, each recognizing a set of objects after training. A sentence schema contains one set of logic variables for the pattern, and another set of logic variables for the weights. Instantiating the schema and evaluating the resulting sentence returns a truth value which is interpreted as the degree of membership of the pattern in the trained set. The "control" part of Kowalski's description corresponds to the mechanism that presents the training patterns and adjusts the values of the weights, using logic variables from the appropriate sets.

Connectionist logic programming (CnxLP) is a restricted version of logic programming (LP), although its control component is more complex. Comparing the two:

- CnxLP programs are schemata for multiple-valued propositional logics; LP programs are Horn clauses in predicate logic,
- CnxLP uses satisfiability as its basic computation mechanism; LP uses resolution,
- CnxLP instantiates single logic variables; LP unifies pairs of logic variables, and
- CnxLP search heuristics are complex (i.e., gradient descent, genetic algorithms, back-propagation); LP uses a depth-first left-to-right search and backtracking.

During 1989 research at Indiana University produced a functioning analog CMOS VLSI processing element for Łukasiewicz' logic, and integrated it into a multiple-valued logic array processor suitable for connectionist logic programming. This processor, called a Łukasiewicz logic array (ŁLA), is a massively parallel analog computer organized as a binary tree of identical processing elements. Each processing element is an 11-transistor cell that performs Łukasiewicz implication (Æ).

Łukasiewicz logic arrays have a dual algebraic and logical operational semantics, thus ŁLAs can implement architectures for fuzzy logic, expert systems and resolution approximation as well as

---

* This is an abstract of the invited talk presented at the 1990 NSF/ICOT Joint Workshop on Parallel Logic Programming and Knowledge Representation, Tokyo, Japan, September 18-21.

* This work was supported in part by the National Science Foundation under the following grants: DCR 85-21497 and MIP 90-10878.

CnxLP processors. The prototype 32-cell ŁLA implementations (versions LL9 and LL10) have been programmed to act as fuzzy function recognizers, the first step in the design of a CnxLP architecture.

Current research includes:

- Designing hybrid digital-analog architectures for CnxLP based on ŁLAs. A digitally programmable "envelop" for ŁLAs and a library of routines to program the ŁLAs from a VAX or UNIX-based workstation are now being tested. The "envelop" allows automated testing of prototype ŁLAs, and replacement of software ŁLAs with real ŁLAs in simulations.

- Measuring the performance of hybrid digital-analog architectures. A series of experiments to determine the Shannon channel capacity of ŁLAs has been designed to run on the ŁLA "envelop." These experiments, together with area measurements of analog and digital ŁLAs, represent the first steps toward determining the information density of equivalent analog and digital systems. Metrics such as information density are needed to justify the design of provably optimal hybrid digital-analog architectures for a given problem domain.

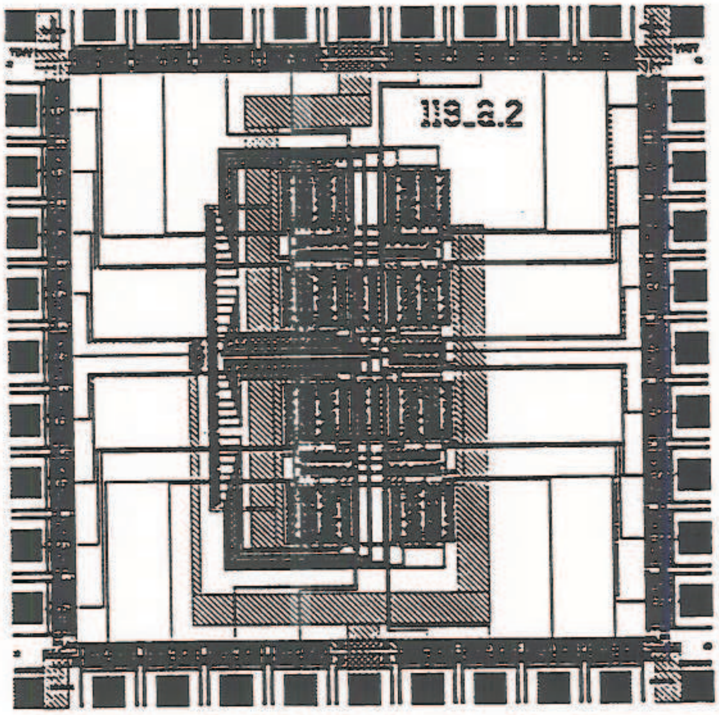
- Developing applications for ŁLAs: neural networks, expert systems and fuzzy logic controllers. While single Hopfield neurons have been simulated by sentence schemata, more efficient use of ŁLAs should be possible. Simple recurrent networks that dynamically recognize tempo are being investigated.

Further study is needed to increase our understanding of programming languages and architectures for hybrid digital-analog processors. Some topics of interest include:

- Programming language design for CnxLP systems,
- Algorithms to derive sentence schemata from neural network descriptions, and
- Complexity, efficiency and optimality of hybrid digital-analog architectures.

Connectionist logic programming provides a restricted — and thus excellent — domain for this research.

Łukasiewicz Logic Arrays



Jonathan W. Mills
Charles A. Daffinger
Computer Science Department

M. Gordon Beavers
Philosophy Department

Indiana University
Bloomington, Indiana USA

Overview

- Origins
- Implementation
- Analysis
- Advantages / Disadvantages
- Programming
- Applications
- Conclusions
- Future Work

Origins

Multiple-Valued Logic

- Lukasiewicz
- Giles
- Zadeh

Cellular Automata

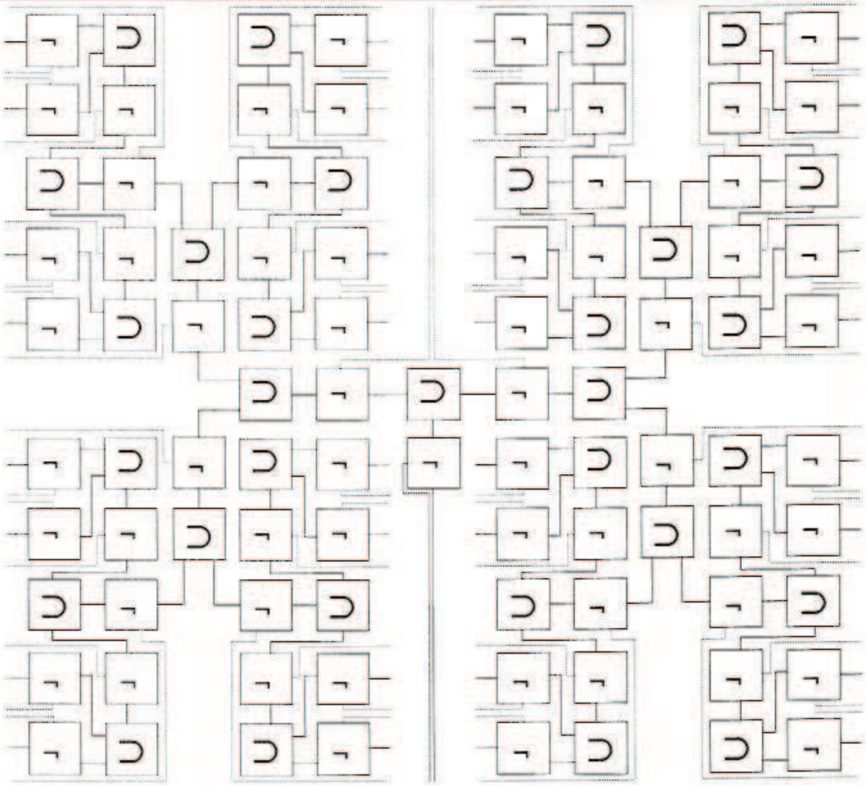
- von Neumann
- Codd
- H. T. Kung

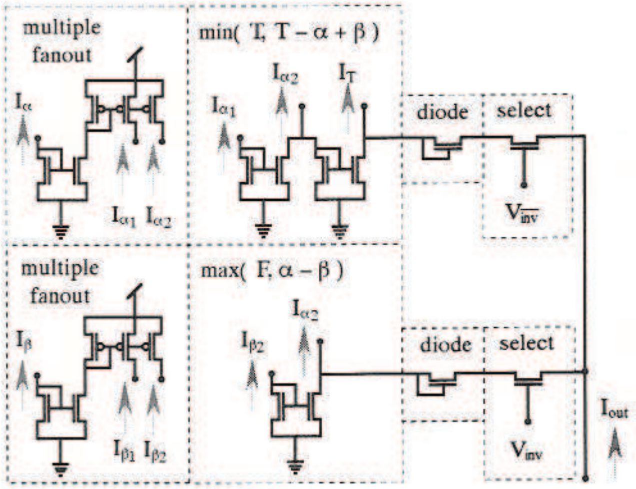
Analog Circuits

- Yamakawa
- Mead

- Cellular automata as computational models for inference systems using multiple-valued and relevance logic
- Synthesis of classical and connectionist AI
  - classical: axiom sets and inference rules
  - connectionist: sentences and valuation functions

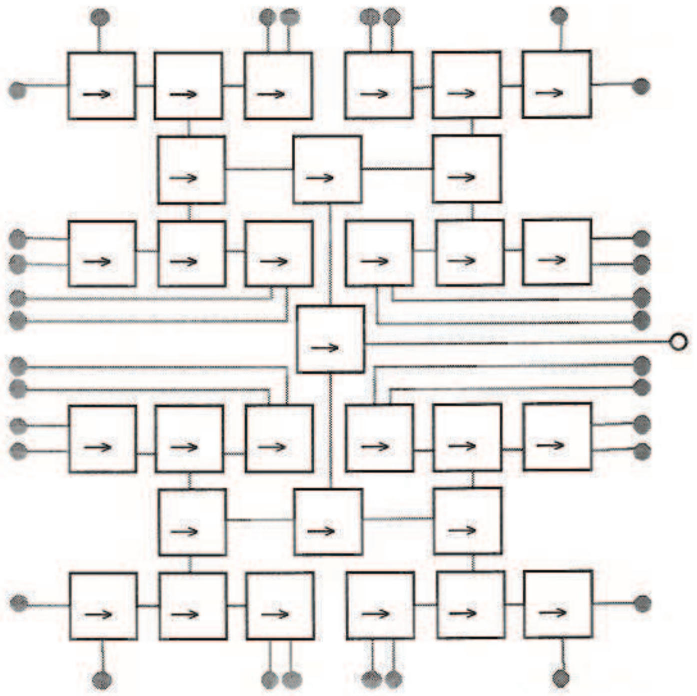
Implementation

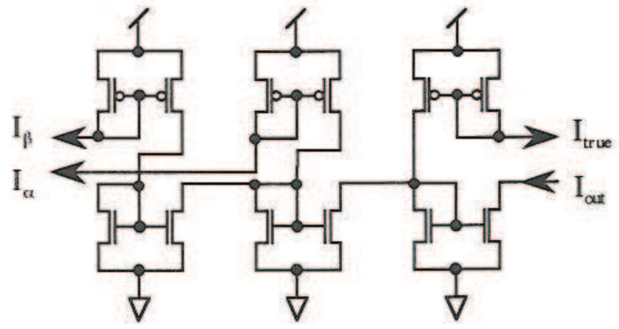



multiple fanout
I_α
I_{α_1} I_{α_2}
min(T, T − α + β)
I_{α_1}
I_{α_2}
I_T
diode
select
$V_{\overline{inv}}$
multiple fanout
I_β
I_{β_1} I_{β_2}
max(F, α − β)
I_{β_2}
I_{α_2}
diode
select
V_{inv}
I_{out}

- First version: H-Tree based on Yamakawa's basic logic cell with selectable implication or negation

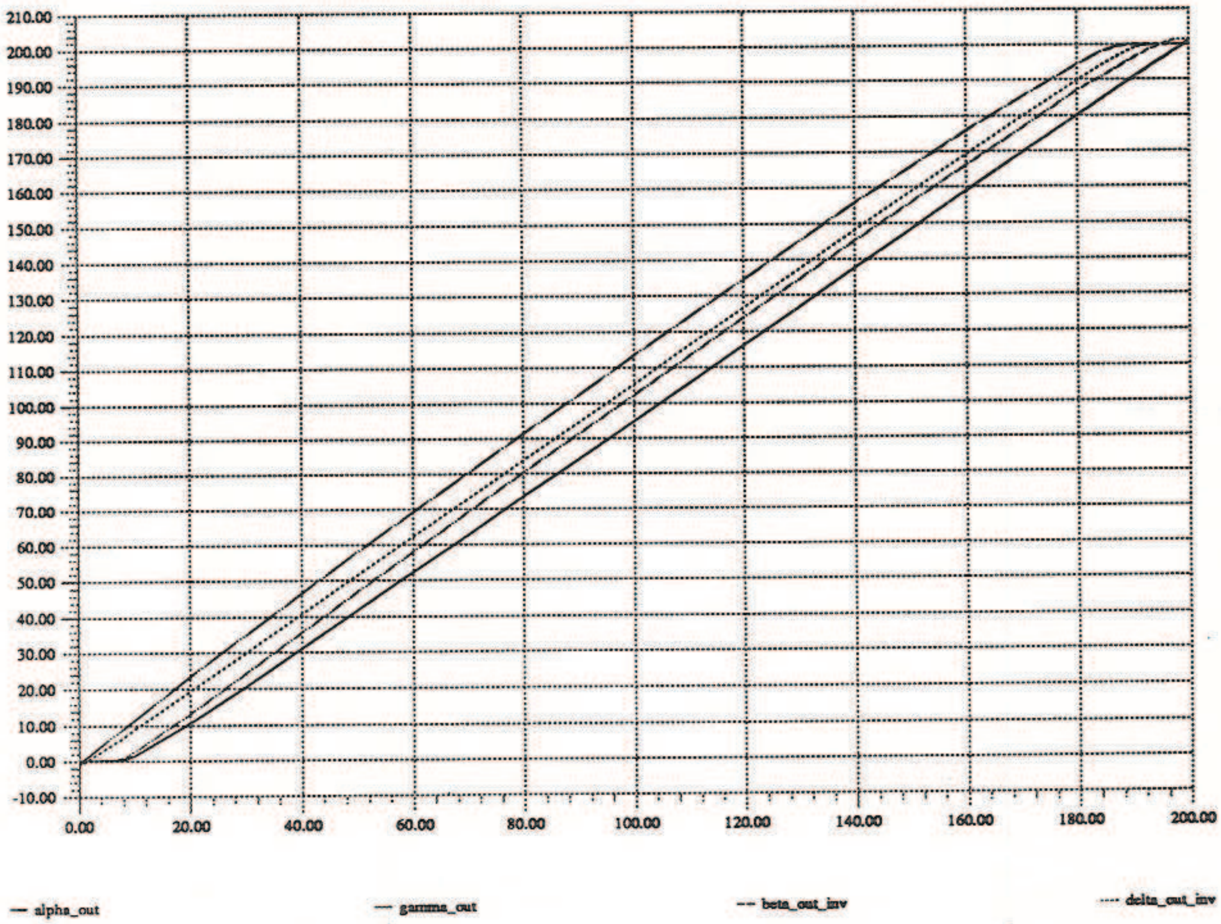
Implementation



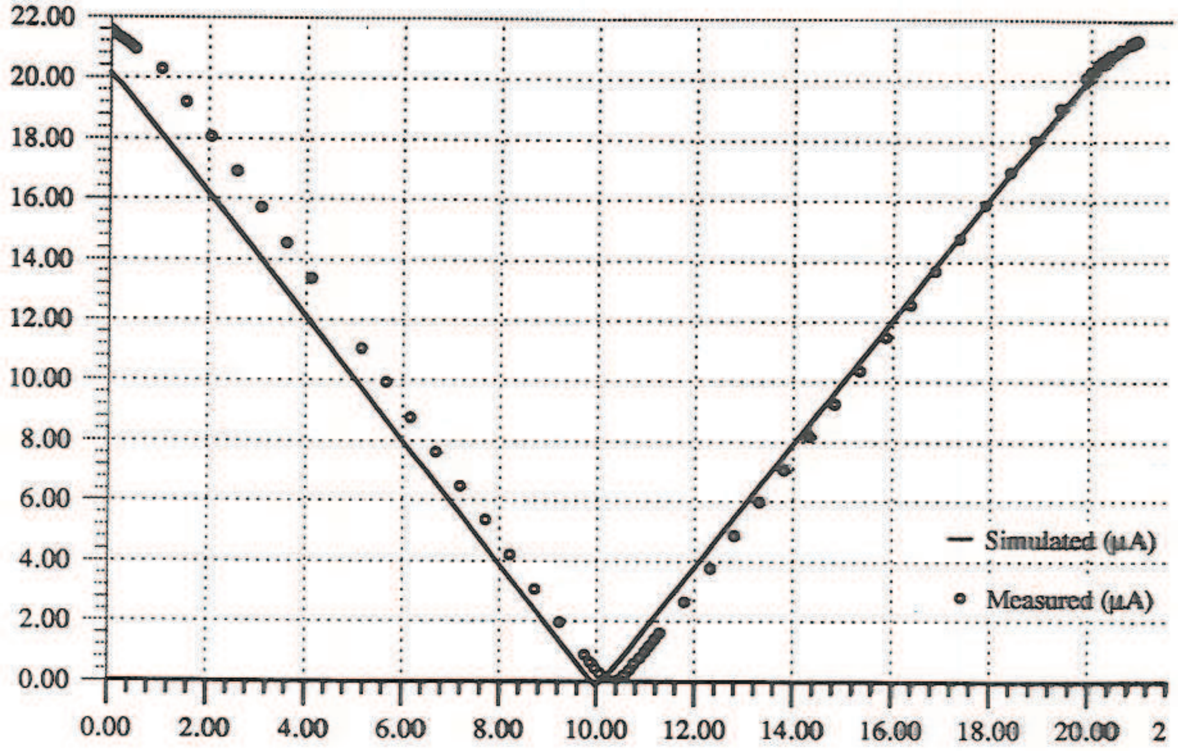

I_β
I_α
I_{true}
I_{out}

- Prototype: H-Tree in implication only

Evaluation of Prototype



Evaluation of Prototype



$$\text{"NOTCH"} \equiv (\neg\alpha \rightarrow \alpha) \rightarrow \neg(\alpha \rightarrow \neg\alpha)$$

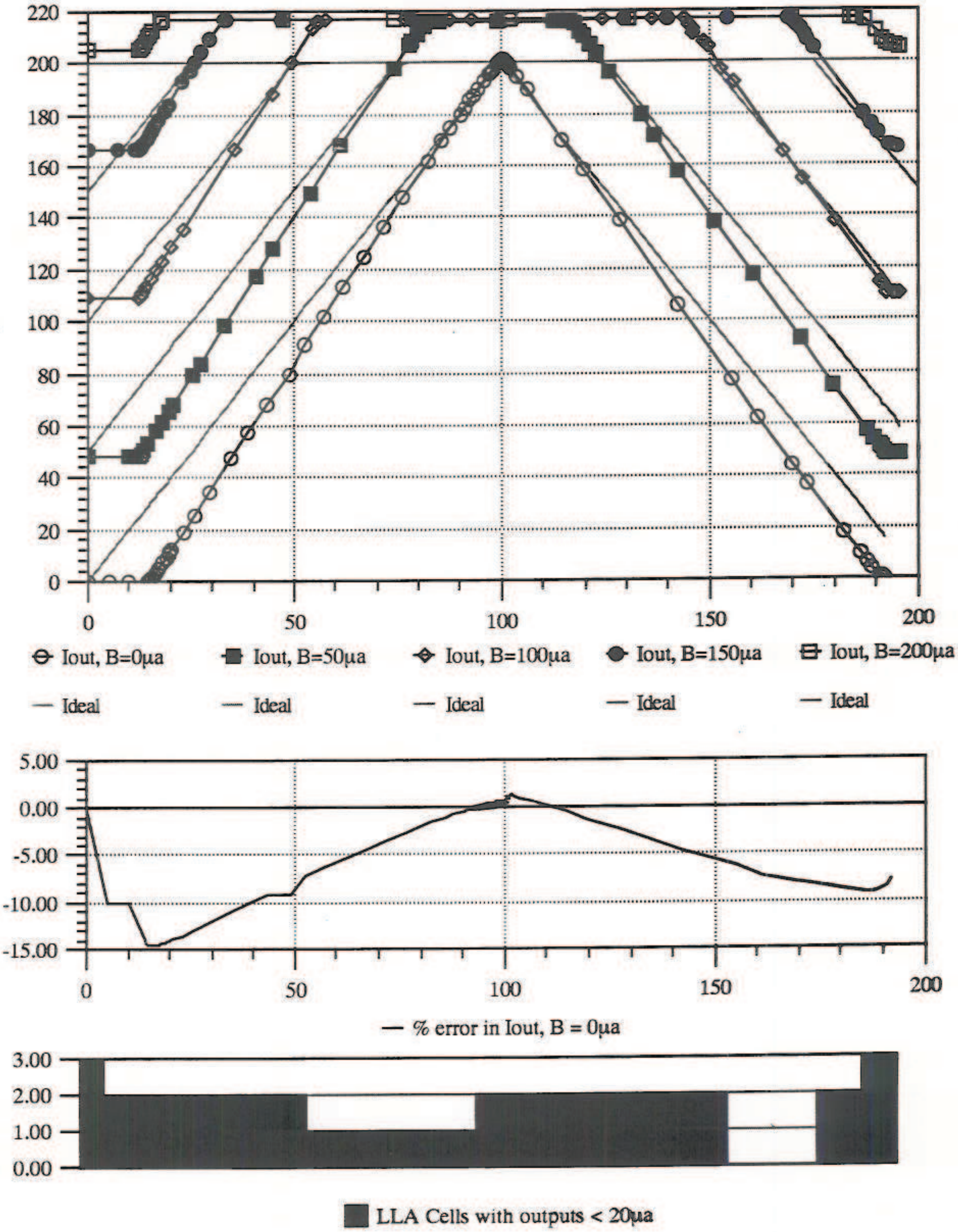


Figure 6. Evaluation of the "notch" using LL9

Advantages & Disadvantages

+ Regular, simple

+ Area-efficient because they are analog (1 wire per signal)

+ Inductive architectures (can be cascaded)

- Precision in the range of 5 to 8 bits for prototype

- Programming introduces data inputs on order $O(2^n)$

- Pin-limited

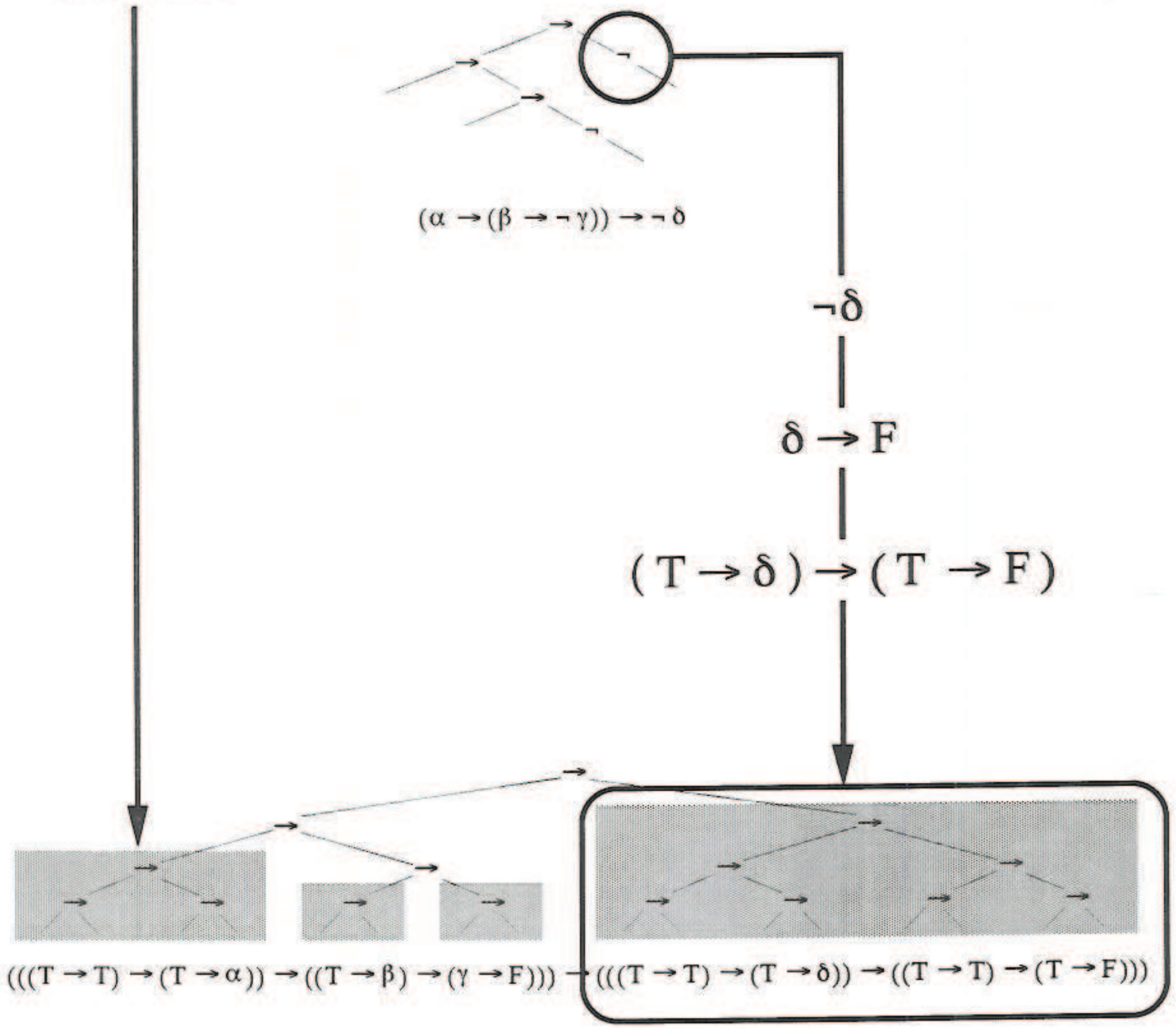
- Difficult to develop programs and predict recursive behavior

- Analog design leads to susceptibility to fabrication error

But! Potential as relatively general-purpose analog processor encourages further investigation

Programming ŁLAs

- Prototype LLA does not have circuitry for negation
- Negation is "unfolded" into implication and logical constant
- Other expansion as necessary to map expression exactly onto LLA



$(\alpha \rightarrow (\beta \rightarrow \neg \gamma)) \rightarrow \neg \delta$

$\neg\delta$

$\delta \rightarrow F$

$(T \rightarrow \delta) \rightarrow (T \rightarrow F)$

$(((T \rightarrow T) \rightarrow (T \rightarrow \alpha)) \rightarrow ((T \rightarrow \beta) \rightarrow (\gamma \rightarrow F))) \rightarrow (((T \rightarrow T) \rightarrow (T \rightarrow \delta)) \rightarrow ((T \rightarrow T) \rightarrow (T \rightarrow F)))$

Applications

- Fuzzy Logic Devices

  - Zadeh (1975) describes fuzzy logic
  - Giles (1976) relates Łukasiewicz logic to fuzzy set theory
  - Yamakawa (1986,1988) multi-function fuzzy controllers

  * fuzzy functions implemented with ŁLA prototype

- Expert Systems

  - Shortliffe (1976) MYCIN expert system
  - Giles (1979) notes relationship of Łukasiewicz logic to Dempster-Shaefer inference

  * decision tree implemented with ŁLA prototype

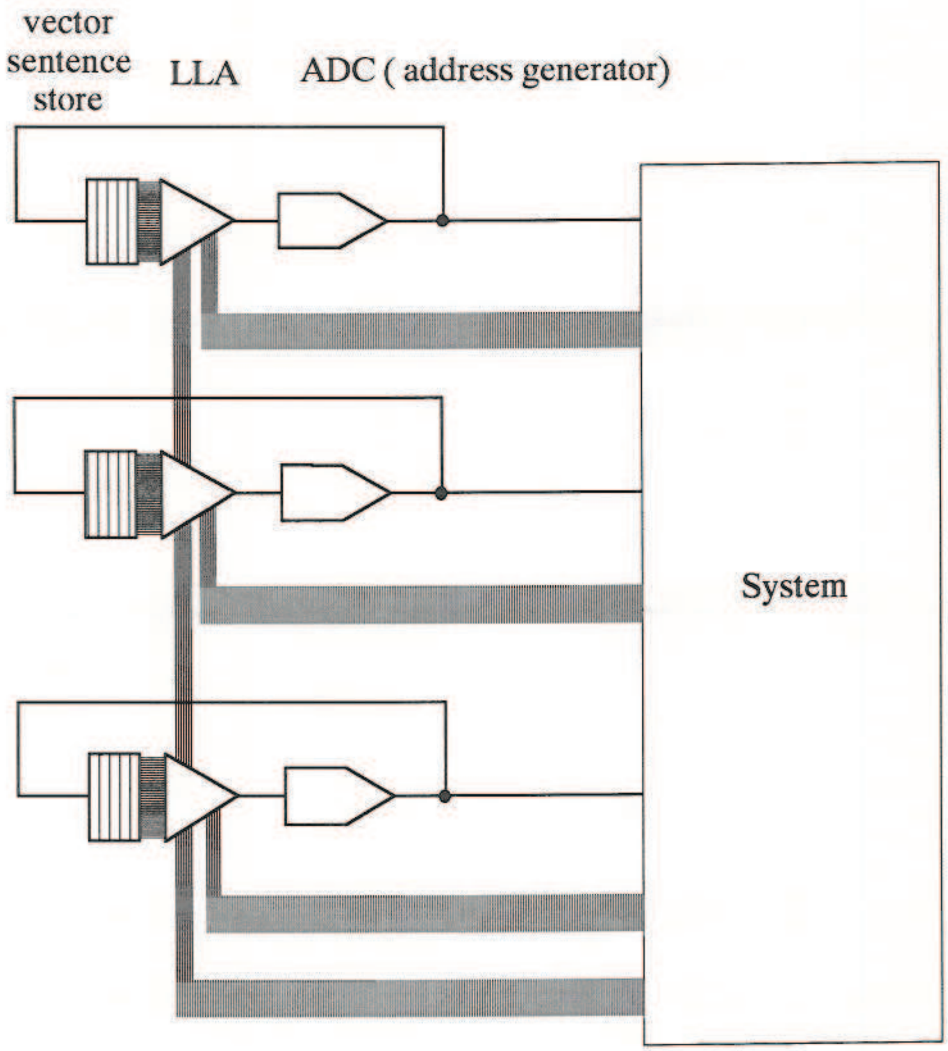
- Neural Networks

  - McCulloch & Pitts (1943) nerve nets
  - von Neumann (1956), Kleene (1956) net automata
  - Alspector & Allen (1987) VLSI neuromorphic system

  * single "neuron" constructed with ŁLA simulator

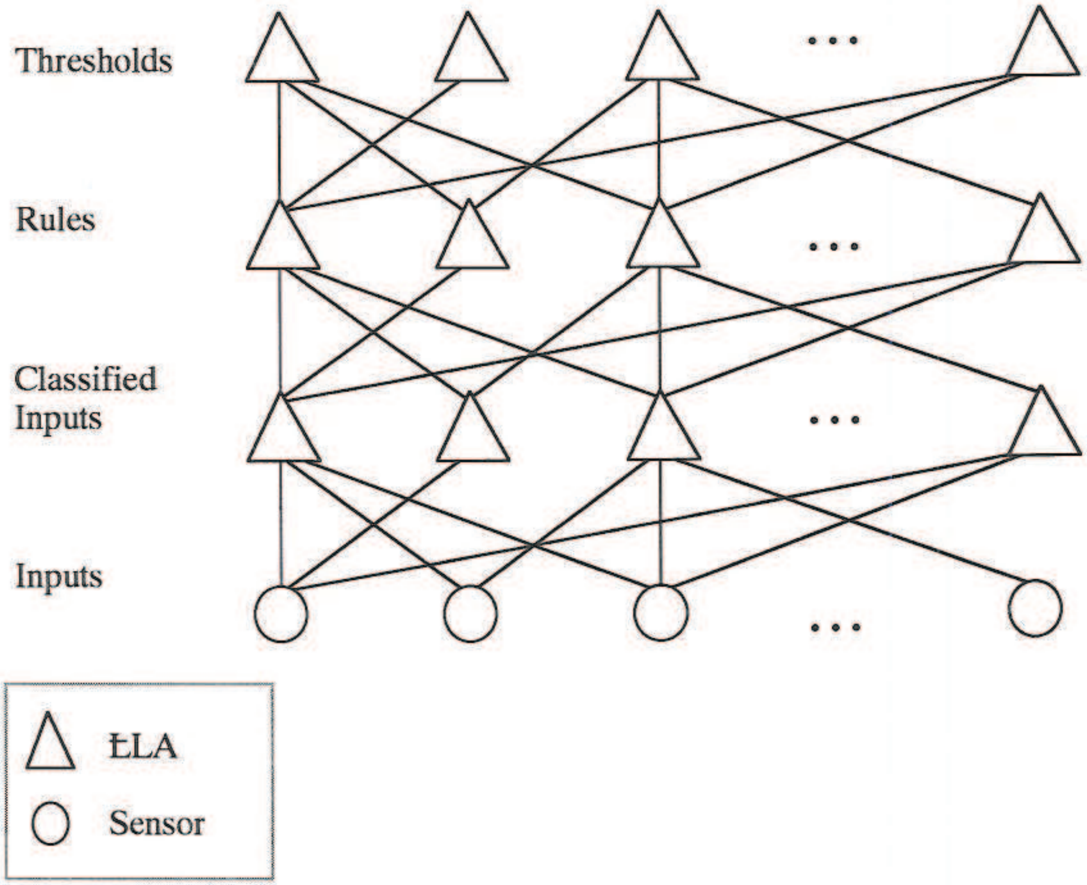
Fuzzy Controller

- LLAs control system functions using different rule sets stored as vectors of logical constants
- Local and global feedback from the system is used to control the system at that time instant and select the next rule


vector sentence store
LLA
ADC (address generator)
System

Expert System

- Inputs may come directly from a transducer, possibly with scaling
- Inputs may be classified based on context (other inputs from transducers or controls)
- Rules are mapped to an LLA
- Threshold units fire when a rule is triggered


Thresholds
Rules
Classified Inputs
Inputs
...
LLA
Sensor

CONNECTING LOGIC TO CONNECTIONISM

McNAUGHTON'S THEOREM (1951)

EVERY SENTENCE S IN Ł HAS A CORRESPONDING LINEAR POLYNOMIAL OVER THE VARIABLES OF S

$$s(u_0, u_1, \dots u_n) \Leftrightarrow p(u'_0, u'_1, \dots u'_n)$$

$\Rightarrow$ S DEFINES A PIECE-WISE LINEAR APPROXIMATION OF SOME DIFFERENTIABLE FUNCTION.

SHANNON – POUR-EL THESIS (1941, 1974)

OUTPUT OF A GPAC (NETWORK, INPUTS, OUTPUT, FUNCTIONS +, −, ×, ÷) CORRESPONDS EXACTLY TO SOME FUNCTION $f(t)$ THAT SATISFIES AN ALGEBRAIC DIFFERENTIAL EQUATION:

$$\sum_{i=0}^{n} k_i f_i^{r_i}(t^{s_i}) = 0 \quad \left| \begin{array}{l} k_i \in \{-1, 0, 1\} \\ r_i, s_i \in \{0, 1 \dots\} \end{array} \right.$$

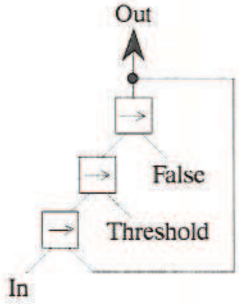
$\Rightarrow$ ŁLA APPROXIMATES GPAC

EXAMPLE: GROSSBERG'S LEARNING EQUATION (1987)

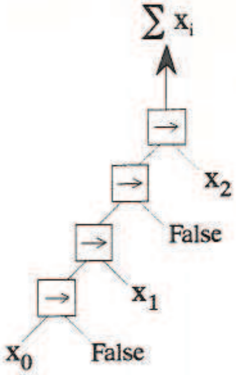
$$\frac{d}{dt} z_{ij} = \left(-z_{ij} + \frac{I_i}{\sum_m I_m}\right) x_{2j}$$

Neural Network

- "Neuron" is constructed in two parts:

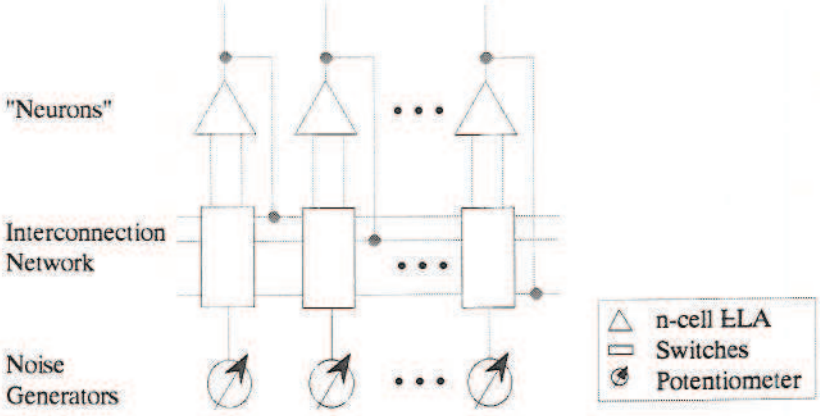

Out
False
Threshold
In

(a) Thresholded output unit

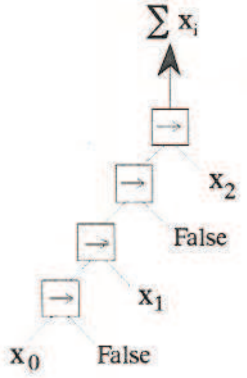

$\sum x_i$
x_2
False
x_1
x_0
False

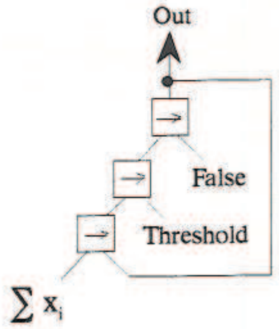
(b) Input summation unit

- At least initially, these units can be mapped to prototype LLA
- Area-efficient neural networks will use a different topology and may mix digital and analog components

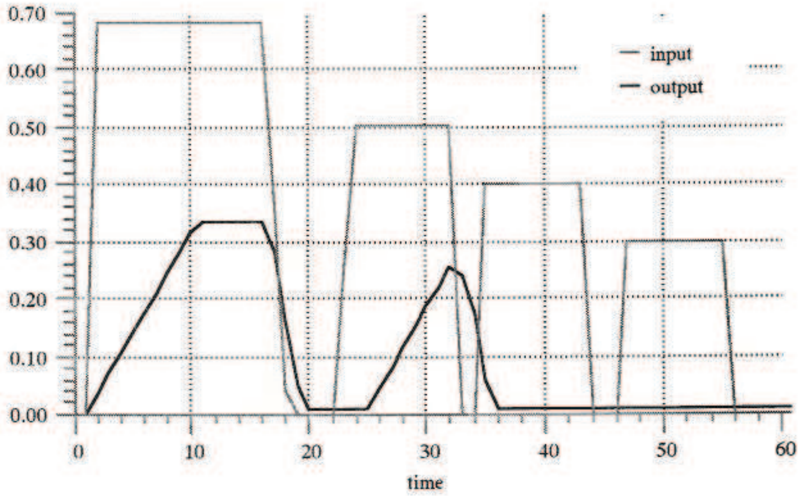

"Neurons"
Interconnection Network
Noise Generators
• • •
n-cell LLA
Switches
Potentiometer

Construction of a Logical Neuron


$\sum x_i$
$\rightarrow$
x_2
False
x_1
x_0
False

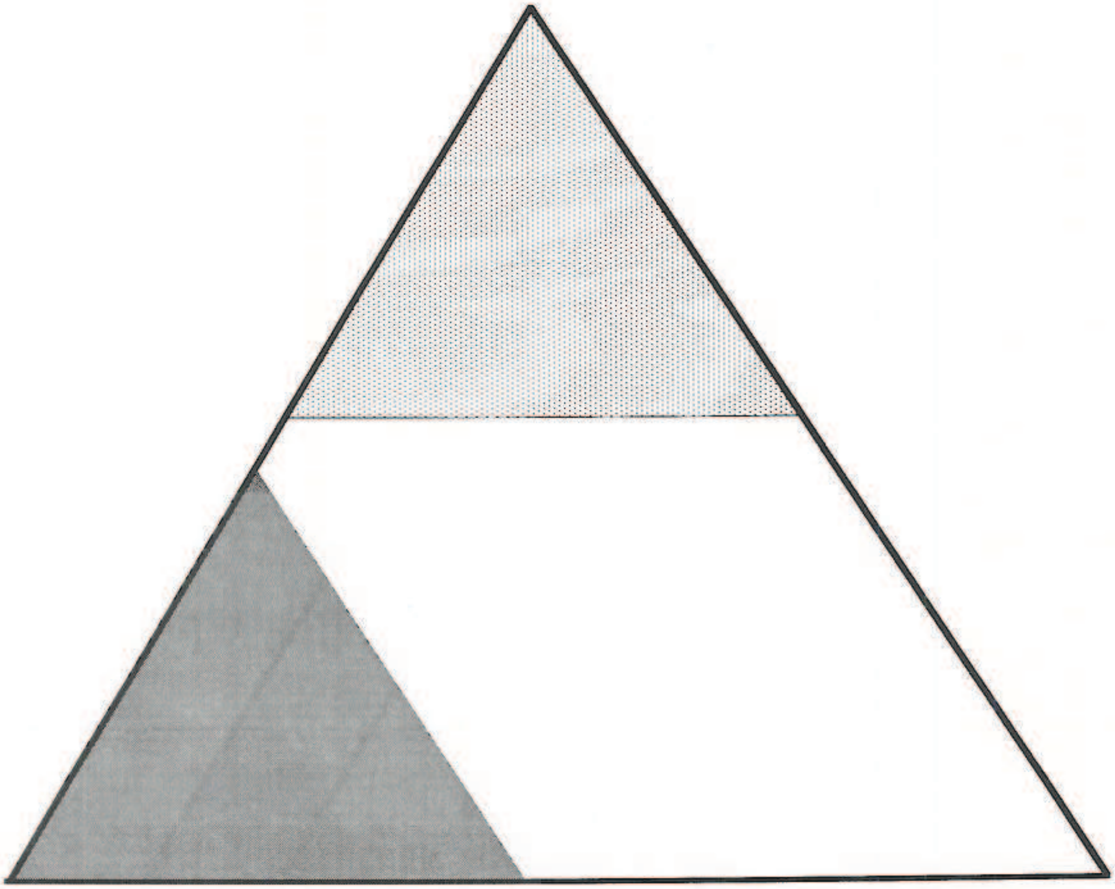

Out
$\rightarrow$
False
Threshold
$\sum x_i$

$$(((\mathrm{In}_t \rightarrow \mathrm{Out}_t) \rightarrow \mathrm{Threshold}_t) \rightarrow F\ ) = \mathrm{Out}_{t+1}$$



$$\min(1,\ 1 - \min(1,\ 1 - \min(1,\ 1 - \mathrm{In}_t + \mathrm{Out}_t) + \mathrm{Threshold}_t)) = \mathrm{Out}_{t+1}$$

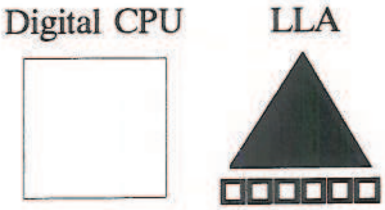
Mixed Interpretations in a Single LLA



Logical (fuzzy inference processor)

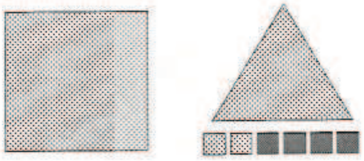
Algebraic (classification network)

CnxLP: Logic Programming on a Hybrid Architecture



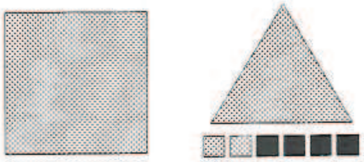
Define the network

```
analog( nn/1).
nn(OutputListN) :- node1( [InputList1], [OutputList1] ),
                   node2( [InputList2], [OutputList2] ),
                                      ⋮
                   nodeN( [InputListN], [OutputListN] ).
```



Train the network

```
train( OutputListN, [InputList1], ... [InputListM] ) :-
             <training procedure>.
```



Run the network

```
                        ⋮
evaluate( OutputListN, [InputList1], ... [InputListM] ),
                        ⋮
```

Conclusions

- An operating CMOS analog VLSI Lukasiewicz logic array was constructed in 1989-1990
- LLAs are current mode devices that operate in the saturation region of MOS FET transistors
- The prototype LLA has 5 to 8 bits of precision
- Errors near the "tail" are due to the design operating too close to threshold voltage (for calculation of$\alpha - \beta$); this error can be rectified by using an additional power supply (Vdd, GND, -Vee)
- LLAs are programmed using vectors derived from sentences in Lukasiewicz logic
- LLAs are "analog PLAs" that can be used to implement fuzzy logic devices, expert systems, and neural networks

Future Work

- Fabricate and test modified prototype
- Test LLAs dynamic behavior
- Implement applications using prototype
  - fuzzy controller
  - expert system
  - neural network
- Design larger application specific LLAs
  - for use as neural networks
  - mixing hybrid digital and analog circuits