# Implementing Valiant's Neuro-Logic 

Jake Beal<br>December, 2005

## How could neural circuits work?

- Many components ( $10^{\wedge} 10$ neurons)
- Large fan-in (24K-80K neighbors)
- Slow cycling (1-10 milliseconds)
- Low precision wiring
- Low synapse strength (0.3-20\% of threshold)


## Valiant's Approach



- Random network with sparse connections
- Symbols are sparse sets of nodes
- Implement circuits by discovering connections


## JOIN(A,B)



- Activate first A, then B
- Nodes activated by both are a new symbol C
- Lower synapse strength so $\mathrm{C}=\mathrm{A} * \mathrm{~B}$


## JOIN(A,B)



- Activate first A, then B
- Nodes activated by both are a new symbol C
- Lower synapse strength so $\mathrm{C}=\mathrm{A} * \mathrm{~B}$


## JOIN(A,B)



- Activate first A, then B
- Nodes activated by both are a new symbol C
- Lower synapse strength so $\mathrm{C}=\mathrm{A} * \mathrm{~B}$


## JOIN(A,B)



- Activate first A, then B
- Nodes activated by both are a new symbol C
- Lower synapse strength so $\mathrm{C}=\mathrm{A} * \mathrm{~B}$


## LINK(A,D)



- Activate A
- Raise synapse strength on D's active inputs
- $\mathrm{A} \rightarrow$ [intermediates $] \rightarrow \mathrm{D}$


## LINK(A,D)



- Activate A
- Raise synapse strength on D's active inputs
- $\mathrm{A} \rightarrow$ [intermediates $] \rightarrow \mathrm{D}$


## LINK(A,D)



- Activate A
- Raise synapse strength on D's active inputs
- $\mathrm{A} \rightarrow$ [intermediates $] \rightarrow \mathrm{D}$


## But does it work?

- Between theory and practice lies engineering
- Simulation feasiblity
- Time
- Device longevity
- Device characterization
- Coding Capacity
- Universality


## Simulation Feasibility

- Even the smallest networks are very large
- 100,000 nodes, 256 links/node, 1981 nodes/symbol
- Bottleneck operations walk all links
- Some tricks I use:
- Links implicitly represented by a random seed
- Two bit-arrays store state
- Generated and cached when necessary
- An arbitrary set is permanently cached
- Fast-fail on link-walk segments with known answers


## Time

- JOIN takes 1 cycle, LINK takes 2
- How can multi-layer logic be synchronized?



## Time

- JOIN takes 1 cycle, LINK takes 2
- How can multi-layer logic be synchronized?
- Buffer C with a JOIN?



## Time

- JOIN takes 1 cycle, LINK takes 2
- How can multi-layer logic be synchronized?
- Buffer C with a JOIN? JOIN(C,C) must fail!



## Time

- JOIN takes 1 cycle, LINK takes 2
- How can multi-layer logic be synchronized?
- Buffer C with a JOIN? JOIN(C,C) must fail!
- Redundant use of inputs in different joins!



## Device Longevity

- JOIN lowers weights, LINK raises weights
- After $x$ JOINs, only $(1-r / n)^{x}$ of a device is usable
- $1,000,000,000$ nodes $=700$ JOINs to destroy a device
- Solution: differentiate into Symbol and Relay nodes

JOIN raises Sym $\rightarrow$ Sym


No Relay $\rightarrow$ Relay

LINK raises Relay $\rightarrow$ Sym

## Device Characterization

- Are they really digital?



## Device Characterization: LINK



- Yes! (Static discipline $\mathrm{V}_{\mathrm{ol}}=0.1<\mathrm{V}_{\mathrm{il}}=0.2<\mathrm{V}_{\mathrm{ih}}=0.5<\mathrm{V}_{\mathrm{oh}}=0.9$ )


## Device Characterization: JOIN



- No! ... and changing the bias turns it into an OR


## Compound AND device



- LINK's nice transfer cleans JOIN's messy output
- Should switch over around 0.85
- Side benefit: no symbol is derived from C


## Chain of ANDs



## Chain of ANDs



- What's going on here?


## Input 0.2



- Why is there a two-stage cycle?
- What are the individual runs doing?


## An anomalous run of input 0.2



- But we're only seeing $2 / 3$ of the time-steps...


## An anomalous run of input 0.2



- But we're only seeing $2 / 3$ of the time-steps...


## An anomalous run of input 0.2



- Spikes at LINK outputs on even timesteps
- Activity from time 0 is rebounding on the links!


## Sources of Crosstalk

- Uniformity of network
- Strong relay activation, sqrt(rn) for a single link
- $3 \%$ of 1,000,000,000 nodes
- Symbol node overlap
- Relay node overlap


## Crosstalk is Cumulative



- Consider 200 independent links, $1 \%$ crosstalk

The net doesn't know about our symbol abstraction!

## Crosstalk is Cumulative



- Consider 200 independent links, $1 \%$ crosstalk

The net doesn't know about our symbol abstraction!

## Crosstalk is Cumulative + Feedback!



- Consider 200 independent links, $1 \%$ crosstalk The net doesn't know about our symbol abstraction!


## Proposed New LINK Algorithm



- Propagate A, marking used relay nodes
- Adjust relay strength as well as D input strength
- $\mathrm{A} \rightarrow \mathrm{E} \rightarrow \mathrm{D}$


## Proposed New LINK Algorithm



- Propagate A, marking used relay nodes
- Adjust relay strength as well as D input strength
- $\mathrm{A} \rightarrow \mathrm{E} \rightarrow \mathrm{D}$


## Proposed New LINK Algorithm



- Propagate A, marking used relay nodes
- Adjust relay strength as well as D input strength
- $\mathrm{A} \rightarrow \mathrm{E} \rightarrow \mathrm{D}$


## Remaining Problems

- Coding Capacity
- Better bounds analysis
- Will the proposed fix work?
- Universality
- Dual rail logic implementation
- Efficient inverter, NAND, etc.
- Feasibility
- Are there more efficient compound gates?

