# Neuron Kanban Cell

## Logical connective implementation [LCI]

Pages suppressed are under CDA.

Colin James III

Ersatz Systems Machine Cognition, LLC

Colorado Springs CO

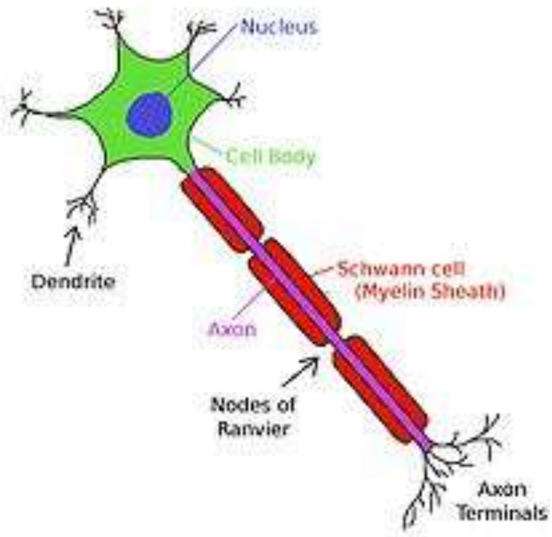
# Table of Contents, 1 of 2

**U = Unpublished (48); P = Published (22)**

# Table of Contents, 2 of 2

# Introduction

# Published Neuron nomenclature




Nucleus = K = Kanban cell

Dendrite = i = input

Axon = κ = output

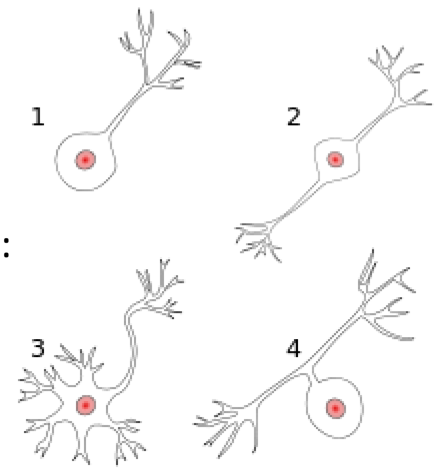
Types of neurons:

1 Unipolar

2 Bipolar

3 Multipolar

4 Pseudounipolar




Neurons have an abstract design in logical gates of a Kanban cell.

The nucleus is the process and feedback places as upper case kappa, K.

A dendrite as input place is i; the axon as output place is lower case κ.

# Published Neuron topology

The input i and output κ is in the form of blocks of four-bits (quadbits).

A signal in quadbits contains two pairs of bits (dibits).

A left dibit is named sinistro; and a right dibit is named dextro.

# Analysis

# Published Analysis of logical gates in a Kanban cell

The Kanban cell is deterministic and bivalent in that it operates on the basis of bit-value:

p as the AND place = 1001;    q as the OR place = 1001.

The quadbit 1001 is viewed as two dibits 10 . 01 for path and content

Sinistro dibits 10. mean: "It is false that the left path is false."

Dextro dibits .01 mean: "It is true that the right content is true".

This Kanban circuitry is assumed, and values of p and q other than 1001 are not optional or entertained.

The input value is i and the output value is κ.

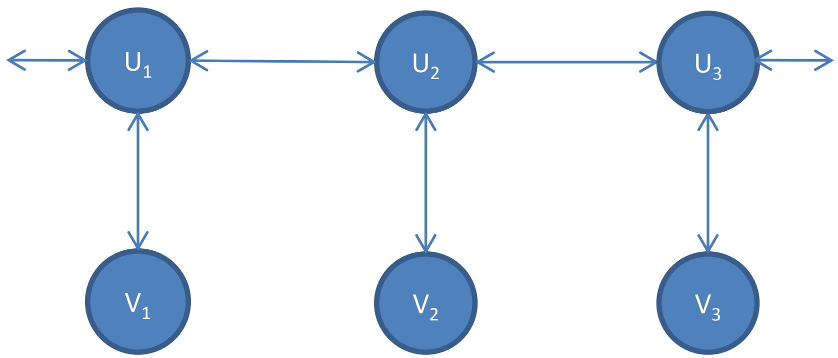
From the formula ( i AND p) OR q = κ, the only variable is i as

( i AND 1001) OR 1001 = κ.

# Design

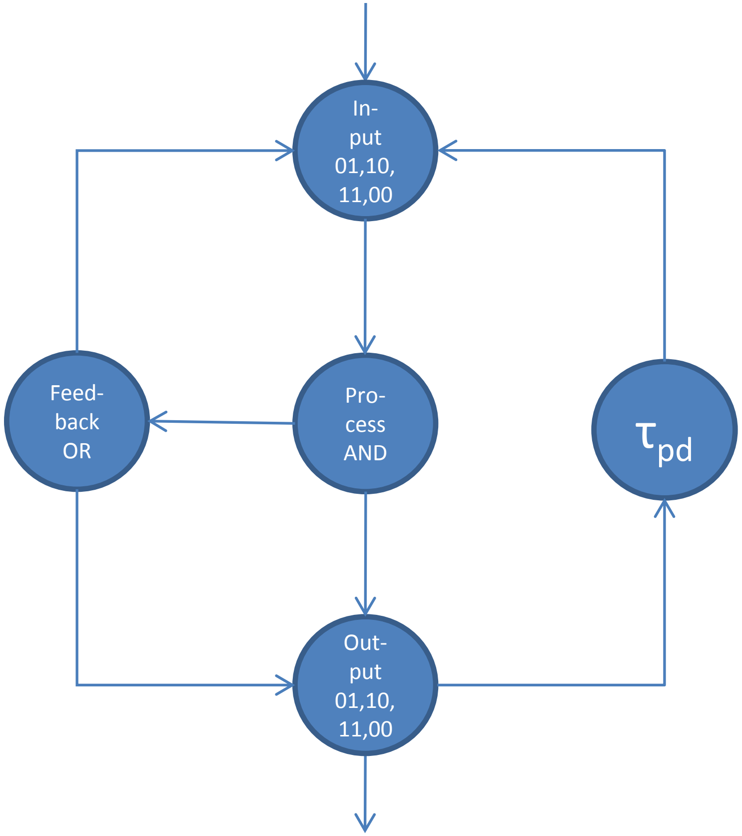
# Published Design of the Kanban cell neuron

This published diagram of a neuron does not work. The U places are a linear shift feedback register (LSFR), and the V places control the flow of U. A clock is absent, but implied for synchronization.

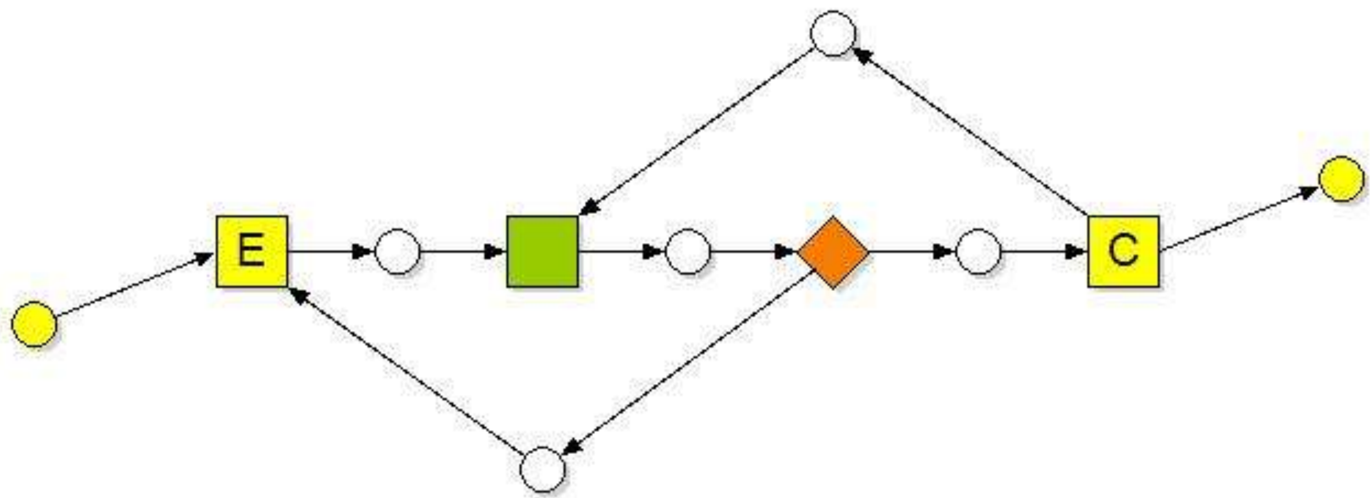


# Preferred embodiment, Example 1: The published design of a neuron core in VHLD for an electrical circuit.

This published Kanban cell does not map a neuron core with feedback from Output to Input, *with* a tau place.



# Petri Net diagram of a unidirectional Kanban cell




Places E and C are Entry and Collection; green square Process is AND; and orange diamond Feedback is OR.

# These logical connectives are defined in the literature.

| Connective | NTAU | AND | NIMP | LP | NIF | RP | XOR | OR |
|---|---|---|---|---|---|---|---|---|
| | Contradiction | Conjunction | Material non implication | Left projection | Converse non implication | Right projection | Exclusive disjunction | Inclusive disjunction |
| | falsehood | and, but, however | but not | p, proposition p | not ... but | q, proposition q | either ... or | or |
| Negation / Opposites | | | | | | | | |
| | Tautology | Alternative denial, non conjunction | Material implication | Left complement | Converse implication | Right complement | Equivalence, biconditional | Joint denial, non disjunction |
| | validity | not both...and | implies, only if, if then | not p, negation of p | if | not q, negation of q | if and only if, just in case | neither...nor |
| Connective | TAU | NAND | IMP | LC | IF | RC | EQV | NOR |

# Published methodology for logical evaluation is:

The formula evaluated is: ( input AND process) OR feedback = output; for input and output, there are four dibit values of 00, 01, 10, 11.

# Published example of pseudo self-timing: NAND and XOR are equal κ

The published Folded Feed-back Kanban (FFK), with Tau replacing Feedback place, is a mistaken design, as based on bidirectional arcs between I→P, I→F, F→O, and P→0.

The two paths of direct exit and the two paths of indirect exit via *mistaken recycling wait* are:

| No. | Direct paths | Indirect paths with wait |
|---|---|---|
| 1, 2 | I → P → O → | I → P → O → I → |
| 3, 4 | I → P → F → O → | I → P → F → O → I → |

# Implementation

# Patent Claims

# Published claim:

… the totality of the apparatus of the invention is named Bi-Valent Neuron (BVN).

# Appendix:

Truth tables, Tabulations, and Implementation Tools

# Derivation of the 16-logical connectives in 4-bits (quadbits) by OR for logical arithmetic

NOT( p) = NOT( p OR p)

NOT( q) = NOT( q OR q)

p NOR q ≡ NOT( p OR q)

p NAND q ≡ ( ( NOT( p)) OR ( NOT( q)))

p AND q ≡ ( NOT( ( NOT( p)) OR ( NOT( q))))

p EQV q ≡ ( ( NOT( p OR q)) OR ( NOT( NOT( p) OR NOT( q))))

p XOR q ≡ ( NOT( ( NOT( p OR q)) OR ( NOT( NOT( p) OR NOT( q)))))

p IMP q ≡ ( NOT( ( NOT( p)) OR q))

p NIMP q ≡ ( ( NOT( p)) OR q)

p IF q ≡ ( p OR ( NOT( q))

p NIF q ≡ ( NOT( p OR ( NOT( q)))

p RC q ≡ ( NOT( NOT( ( NOT( p)) OR ( NOT( q))) OR q))

p RP q ≡ ( NOT( ( NOT( p)) OR ( NOT( q))) OR q)

p LC q ≡ ( ( ( NOT( p OR q)) OR ( NOT( NOT( p) OR NOT( q)))) OR q)

p LP q ≡( NOT( ( ( NOT( p OR q)) OR ( NOT( NOT( p) OR NOT( q)))) OR q))

p TAU q ≡ ( ( ( NOT( p)) OR ( NOT( q))) OR q)

p NTAU q ≡ ( NOT( ( ( NOT( p)) OR ( NOT( q))) OR q))

# Derivation of the 16-logical connectives in 4-bits (quadbits) by AND for logical arithmetic

NOT( p) = NOT( p AND p)
NOT( q) = NOT( q AND q)

| | | |
|---|---|---|
| p NAND q | ≡ | ( NOT( p AND q)) |
| p OR q | ≡ | ( NOT( ( NOT( p)) AND ( NOT( q)))) |
| p NOR q | ≡ | ( ( NOT( p)) AND ( NOT( q))) |
| p EQV q | ≡ | ( ( NOT ( p AND ( NOT( q)))) AND ( NOT ( ( NOT( p)) AND q))) |
| p XOR q | ≡ | ( NOT( ( NOT ( p AND ( NOT( q)))) AND ( NOT ( ( NOT( p)) AND q)))) |
| p IMP q | ≡ | ( NOT( p AND( NOT( q)))) |
| p NIMP q | ≡ | ( p AND( NOT( q))) |
| p IF q | ≡ | ( NOT( ( NOT( p AND q)) AND ( NOT( q)))) |
| p NIF q | ≡ | ( ( NOT( p AND q)) AND ( NOT( q))) |
| p LC q | ≡ | ( ( NOT( p AND q)) AND ( NOT( q))) |
| p LP q | ≡ | ( NOT( ( NOT( p AND q)) AND ( NOT( q)))) |
| p RC q | ≡ | ( NOT( ( ( p AND ( NOT( q))) AND ( NOT( ( NOT( p)) AND ( q)))) AND ( NOT( q)))) |
| p RP q | ≡ | ( ( ( p AND ( NOT( q))) AND ( NOT( ( NOT( p)) AND ( q)))) AND ( NOT( q))) |
| p TAU q | ≡ | ( NOT( ( p AND q) AND ( NOT( q)))) |
| p NTAU q | ≡ | ( ( p AND q) AND ( NOT( q))) |