NeurOn Neural Network Design Language and Compiler

1.0.0 Alpha



User Manual

Contents

* About neurOn
* Command Line Arguments
* Compilation Chain
* Source files and Object files
* Syntax
  + Expressions
  + Groups
  + Lists
* Operators
* Variables
* Macros
* Expressions
* Functions
* Inline API
* Dependencies
* Makefile compilation

About neurOn

Biological neural systems are extraordinarily complex. In a SAPA system, the intention is to create a sort of hybrid logical system that behaves similarly to a biological model but built to function on modern hardware. The intent is to create programs capable of complex learning.

With rapid development in mind, it can be extremely tedious and time consuming to constantly write, re-write and edit lower level code. Thus, neurOn offers a higher level way of designing systems without worry about the underlying overhead.

The principle is similar to a hardware design language such as VHDL or Verilog. Circuits may be designed via written expressions and compiled into a final product. However, there are several major differences.

First of all, SAPA systems are dynamic. The program written and compiled will only yield an optimized program and a starting point. From there, a system is free to evolve and learn as time progresses. Secondly, the systems intended to be created are significantly more complex than an average electrical circuit. Thus, expressions are designed with extreme complexity in mind and make heavy use of loops and regional generations.

The intent of this part of the SAPA toolset makes the development of learning machines quick and (relatively) simple. The remainder of this manual contains definitions, explanations and examples to begin creating unique neural systems.

Command Line Arguments

The table below lists all accepted command line arguments. Note that options must be set in the form of -[n] or --[setting]=*value* with no spaces in an expression.

|  |  |
| --- | --- |
| -v | Print version |
| -h | Print Help Dialogue |
| -o | Activate optimization |
| -d | Debug mode |
| -e | Enable evolution |
| -w | Suppress warnings |
| -s | Preserve C source code |
| -c | Compile to object and C files, do not compile to binary |
|  |  |
| --bitmode=*32/64* | Set to 32 or 64 bit compilation |
| --output=*executable* *output directory* | Directory to output build |
| --title=*name* | Set build name |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Syntax

In order to maintain readability, neurOn is designed with a block scheme. Objects, such as regions and individual components are declared and then initialized within their block. There are some exceptions of course such as aliasing and pre-processing statements. The following paragraphs each explain a particular structural concept.

Comments:

Comments can be single or multi line. Single line comments start and end a comment with the ‘#’ character. Although a comment will terminate at the end of the line, it may be ended sooner. For example: *This circuit is #probably# not going to create a terminator.*

A multi-line comment is similar, but used “#\*” to begin a comment and “\*#” to end one.

Aliases:

An alias may take the place of a more complex expression or tedious name. An alias is defined with the *pseudo* command. The syntax is *pseudo expression: alias*. A multi-line alias may be defined within braces. For example:

pseudo {“This is a

multi line alias that

doesn’t make the compiler angry”}:alias

Cell Definitions:

A cell is an object that inherits the attributes of the

Region Definitions: