

# Spiking neural networks software

---

## User guide

### 1. Launching software

Software requires Microsoft .NET 3.5 Framework installed. When application is launched a Getting Started window (Figure 1) appears. In this opening window, the user can select if he wants to open an existing network structure, or create a new workspace where the new structure will be designed. An existing network can be selected from the list of the most recent projects which will appear in a white *listbox* control, or it can be opened by the browse button.

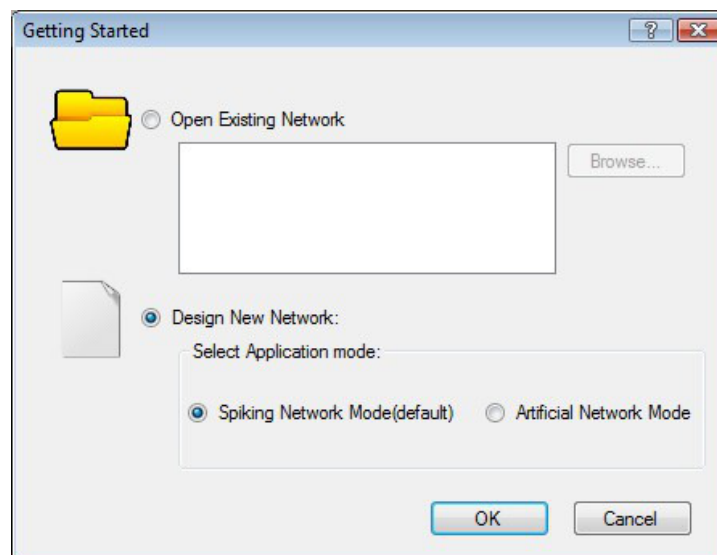


Figure 1. Getting Started window

### 2. Basic functions overview.

In addition to implementation of spiking neural networks, the software application has many useful features to present network behaviour, design network structure and generate hardware description language code (VHDL). The main application window has a menu strip of which items hierarchy is as follows:

- File – holds basic application commands responsible for creating new project or opening/saving network structures, including:
  - New – to start new project, opens new workspace where new network structure can be designed.
  - Load Structure – to load existing network structure from INI file.

- Save Structure – to save designed structure into INI file.
- Application Mode – allows to select application mode:
  - Spiking Networks – mode for spiking neural networks
  - Artificial Networks – mode for traditional artificial neural networks
- Recent Projects – holds list of most recent projects
- Exit – to close application
- View – holds commands to show or hide windows and tool boxes.
  - Structure ToolBox – to show or hide Structure tool box
  - Windows
    - Workspace – to show or hide workspace window
    - Neuron Details – to show or hide Neuron Details window
    - Simulation Initialization – to show or hide Simulation Initialization window
    - Simulation – to show or hide Simulation window
- Simulation
  - Initialize Simulation – To initialize and begin simulation process, once this command is executed Simulation Initialization window appears and initialization process can begin.
- Tools – holds commands to launch various application tools useful for designing SNN.
  - Plot Spikes – allows plotting spike raster plots for individual layers and for entire network.
    - Input Layer
    - Layer 1
    - Output Layer
    - Entire Network Activity
  - Gaussian Receptive Fields – opens Gaussian Receptive Fields tool, which shows this method of coding and allows making hypothetical calculations.

- Input File Generator – opens tool for generating text files with patterns encoded for pattern recognition.
- Generate VHDL Code – generates VHDL code of designed network.
- Options – opens application options window to set various options.
- Help
  - Contents – opens main application help.
  - About – opens window with basic information about application

In addition, a tool strip has been added to provide easy access to the most frequently used commands like: save, open or create new project. This control also shows current simulation time<sup>1</sup>. This tool strip is shown in Figure .

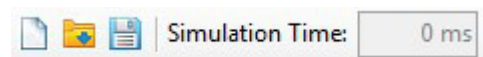


Figure 2. Application Tool Strip

## 2.1. Network structure editor toolbox

One of the basic functions of the application is to create structure of spiking neural network and this can be done by using toolbox shown in Figure 3.

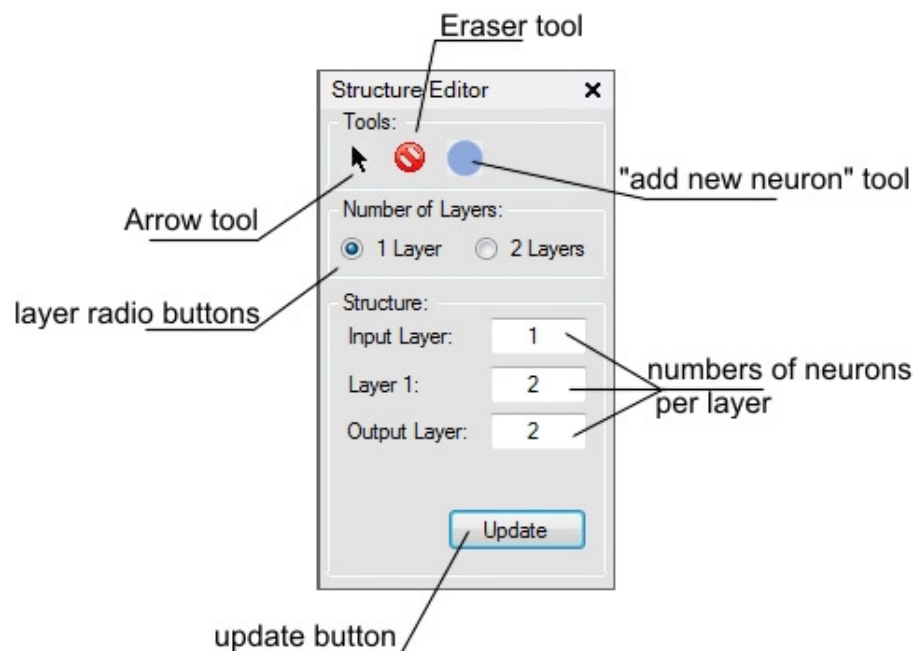


Figure 1. Structure Editor toolbox

<sup>1</sup> Taking into consideration time scale mentioned in chapter Error: Reference source not found it does not show real time value.

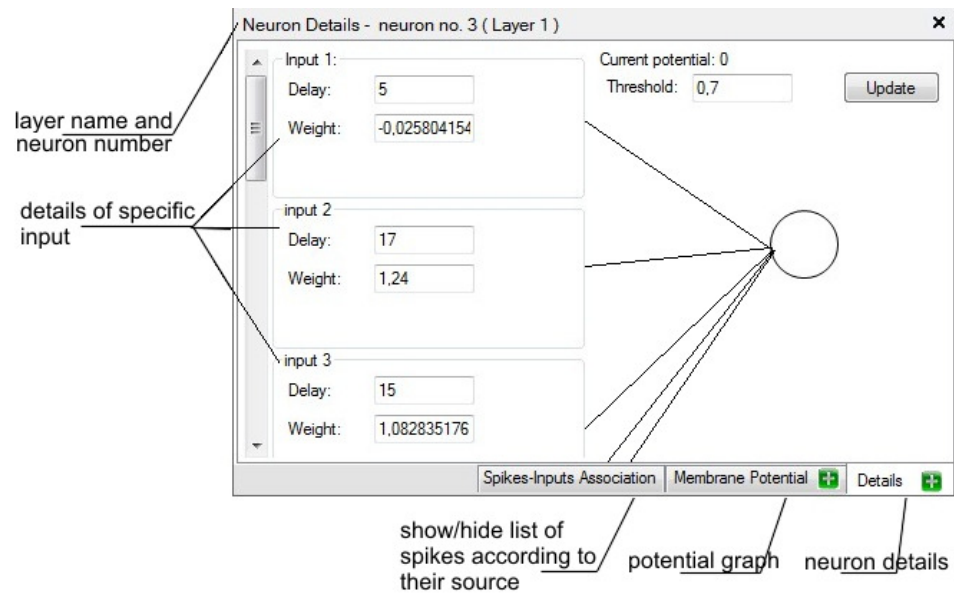
There are two ways to design a neural structure. The first way is to use *TextBoxes* that corresponds to a particular layer. One should just enter desirable values into them and press the *Update* button. The other way is to use 'add new button' tool, which after is chosen, can be used to add a neuron instance of particular layer on workspace window. When user has chosen this tool and clicks on the workspace area, context menu appears where he can select neuron of which layer will be added and then enter parameters for this neuron. Two radio boxes in *Number of layers* section are used to determine how many hidden layers designing network will have, but as it was mentioned before only one hidden layer is working.

After neural network structure has been created, user can use an '*arrow tool*' to move neurons on the workspace window. One neuron or group of neurons can be moved. To group neurons, user has to click on a workspace and drag mouse to select neurons. Selected neurons will be highlighted on yellow. To unselect neurons user has to click again workspace area. Click on single neuron opens window where the neuron details can be seen and moreover membrane potential can be observed. Right click on single neuron shows context menu where user has easy access to the neuron variables, like delays or weights, and these values can be edited through this context menu. Changes are inserted after the *Update* button is clicked.

By using an *Eraser* tool, user can remove any neuron from neural structure. Once neuron has been removed this operation can't be undone.

## **2.2. Neuron details window**

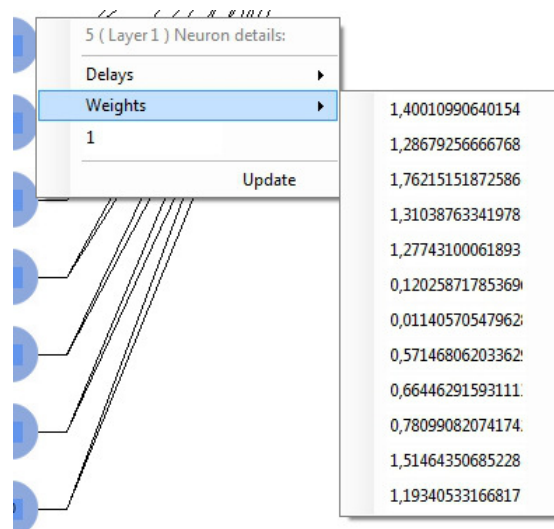
This part of application shows details about the selected neuron. Using it we can modify neuron parameters like its threshold, input delays or weights. Values are grouped according to the specific input. Within a header of this window one can see what is the number of selected neuron and to which layer it belongs to. During the simulation current membrane potential is shown. At the bottom of this window there are three tabs. The first tab holds the neuron details, second for graph membrane potential and third tab is in fact a button that shows or hides list of dependencies between spikes and inputs.



**Figure 2. Neuron Details window**

### 2.3. Neuron details context menu

When one click a neuron with the right mouse button, a context menu appears and gives quick access to the neuron details. On the top of this menu one can see simple information about layer and neuron number, two submenus with input delays and weights. Every value is an inside textbox control and can be easily modified. After changes done there is an update button to submit changes. This menu is shown in Figure 3, where the details for 5<sup>th</sup> neuron from a hidden layer 1 are shown. Delays and weights are easily accessible and so is threshold which in this case is set to 1. On the bottom of this context menu we can see an *Update* button which after is clicked, invokes the method to approve changes so neuron parameters are updated.



**Figure 3. Context menu of neuron details**

### 3. Advanced functions overview.

#### 3.1. Membrane Potential plotter

This element is responsible for drawing the membrane potential according to values stored in a network. The entire graph can be drawn in black but also in colours according to the neuron inputs. Following example is made to explain this element and to prove that potential is calculated correctly and delays of synapses are really taken into consideration. A simple network with two input neurons, one hidden layer with one neuron within it, and one output neuron is created (Figure 6).

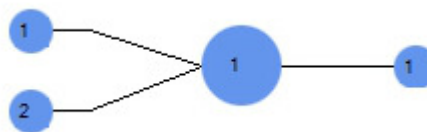


Figure 4. Network used to present how potential plotter works.

For the demonstration, to present a functionality of the Membrane Potential Plotter all weights has been set to 1 and a delay of synapse between input neuron number 1 and a hidden layer neuron is equal to 100ms, while a synaptic delay between input neuron number 2 is equal to 550ms. This significant difference between delays is to show that the synapse really delays spike according to introduced model. Both input neurons fires a spike every 100ms and send it though synapse. One can examine how the membrane potential for hidden layer neuron behaves, thus connection to output neuron is not important. A *TimeScale* variable is set to 2ms. The *test* simulation mode has been selected. Around 800ms of simulation is recorded.

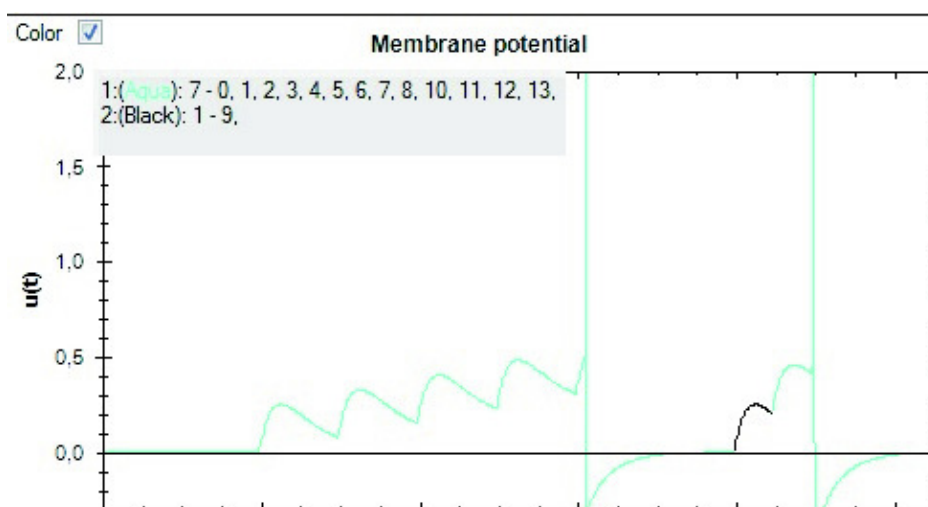


Figure 5. Hypothetical membrane potential graph

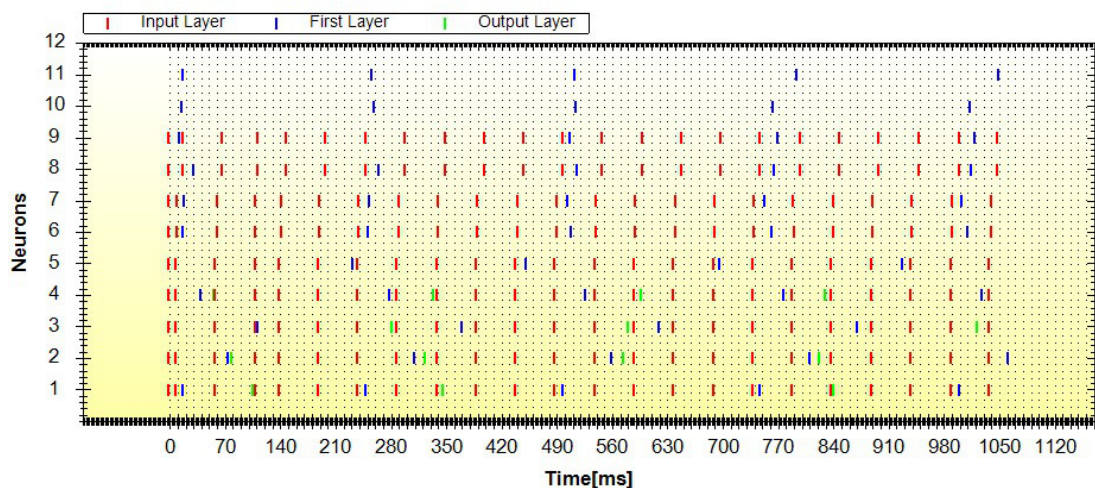
In a table below an input spikes firing times has been shown and its arrivals to postsynaptic neuron in hidden layer 1 corresponding to the foregoing example.

**Table 1. Time and connection parameters for tested network.**

Input	Spike index	Spike fire time	Synapse delay[ms]	time when PSP occurs[ms]
Input 1	S1	100ms	100ms	200ms
	S2	200ms	100ms	300ms
	S3	300ms	100ms	400ms
	S4	400ms	100ms	500ms
	S5	500ms	100ms	600ms
	S5	600ms	100ms	700ms
Input 2	S1	100ms	550ms	650ms
	S2	200ms	550ms	750ms
	S3	300ms	550ms	850ms
	S4	400ms	550ms	950ms
	S5	500ms	550ms	1050ms

### 3.2. Spike history plotter tool (spike raster plot)

From the *Tools* section in an application menu, one can go to the *Plot Spikes* section and select desired layer. A new window will be opened where the history of every spike that occurred within the simulation time and sum of spikes for every millisecond is shown. An application provides the spike raster plots for every layer: input, hidden layer 1, and the output layer, which plot spikes in one colour among the simulation time only for chosen layer. In addition, by using an *Entire Network Activity* tool, user can plot spikes for all layers in one chart.



**Figure 6. Spike raster plot example**

### 3.3. Gaussian Receptive Fields tool

This tool is designed to be helpful in understanding the Gaussian Receptive Fields encoding. Main part of this tool is a chart on which Gaussian functions for every field is drawn. In *Parameters* area user can define encoding parameters like the  $\gamma$  constant, number of fields, or range of encoding. The *Maximum time* field sets value for maximal firing time of input neurons, minimum is by default set to 0 and it means that neuron will not fire. A minimum value can not be changed. After a *Draw* button is clicked tool draws curve function for the every single receptive field. In the *Encoding* area, we can see a textbox to insert value to encode, and after *Calculate* button is clicked all the calculations are being done. In addition, a thin vertical bar is placed on a chart to indicate position of encoding value so the points of intersection can be easily seen.

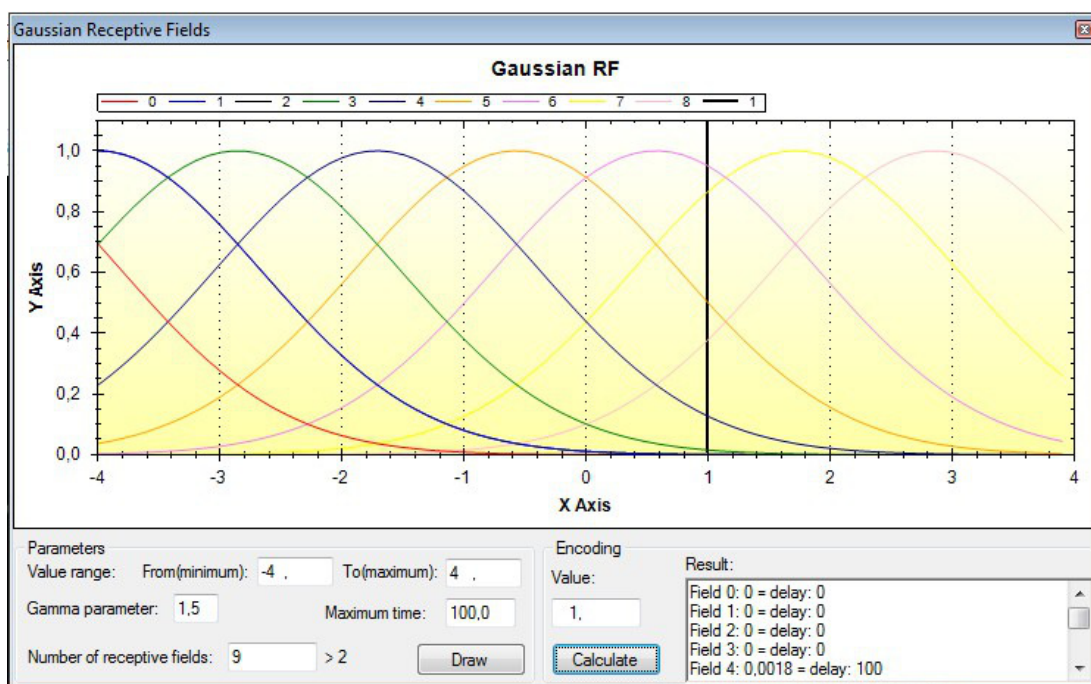


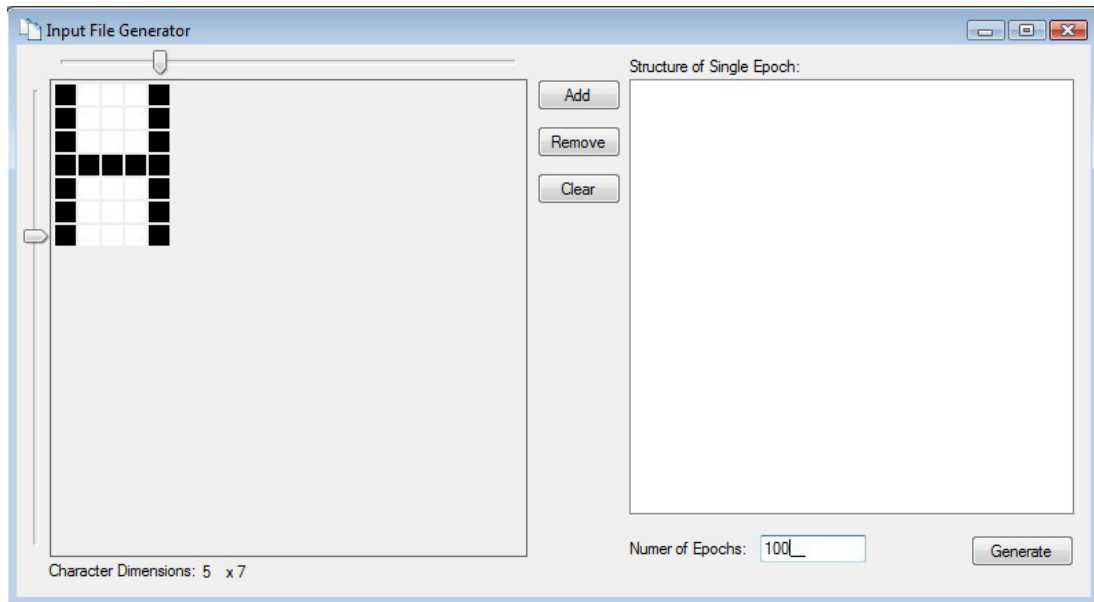
Figure 7. Gaussian Receptive Fields tool

In Figure 9 the encoding of value 1.0 within range  $\langle -4.0; 4.0 \rangle$  and with 9 receptive fields is shown. In Result box user can see the Gaussian function values for every field and also neuron delays encoded in milliseconds, where 0 for delay means that neuron will not fire.

### 3.4. Stimulus File generator

This tool can encode any pattern created with clickable squares on the left side of the form into single code word composed of 0 and 1. This tool can only generate stimuli files for pattern recognition and classification purpose.





**Figure 8. Stimulus file generator.**

To create a file, one should adjust desired dimension of one pattern by using vertical and horizontal *trackbar* controls on the left side of form. Moving *trackbars* change number of clickable squares on the panel. Every square when is clicked can change colour between black and white, and when code is being created, every black square is represented by 1 while every white one by 0. By pressing *Add* button user adds encoded patterns and creates representation of one single epoch within a file, then in *Number of epochs* field can determine how many epochs created above fill will have. “*Generate*” button allows to save epochs to a text file. In addition buttons *Remove* and *Clear* has been provided to make creating file easier. Figure below presents pattern encoding into one word 1001011001101001, which illustrates coding method used by Stimulus File generator.

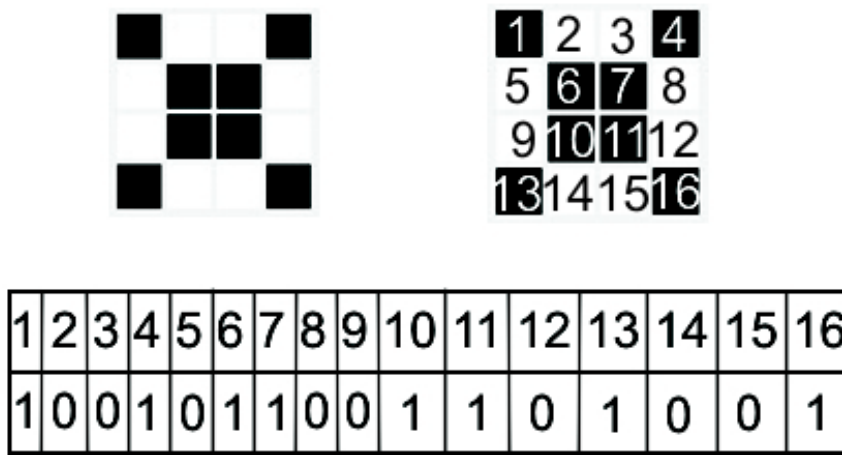


Figure 9. Example of pattern encoding.

### 3.5. VHDL code generator

This module is responsible for creation of the VHDL files of designed network. Once the network structure is created user can generate a hardware model which later can be synthesized onto FPGA board.

Process of VHDL code generation is based on the source files of hardware implementation stored in `../hardware/SNN/src/` folder. These files include a sort of tags placed instead of network parameters, the generation process parses files and every tag is being replaced by a value corresponding to it according to the information stored in `snnTags.dat` file. This solution allows generation process to be very flexible because every tag from a file mentioned above is being replaced by other string corresponding to it. In this connection, if the source files will be changed tag file should be change adequately so the other model of network can be generated. Some tags with their description are shown in Table 1, and if not mentioned otherwise every value is stored in binary code.

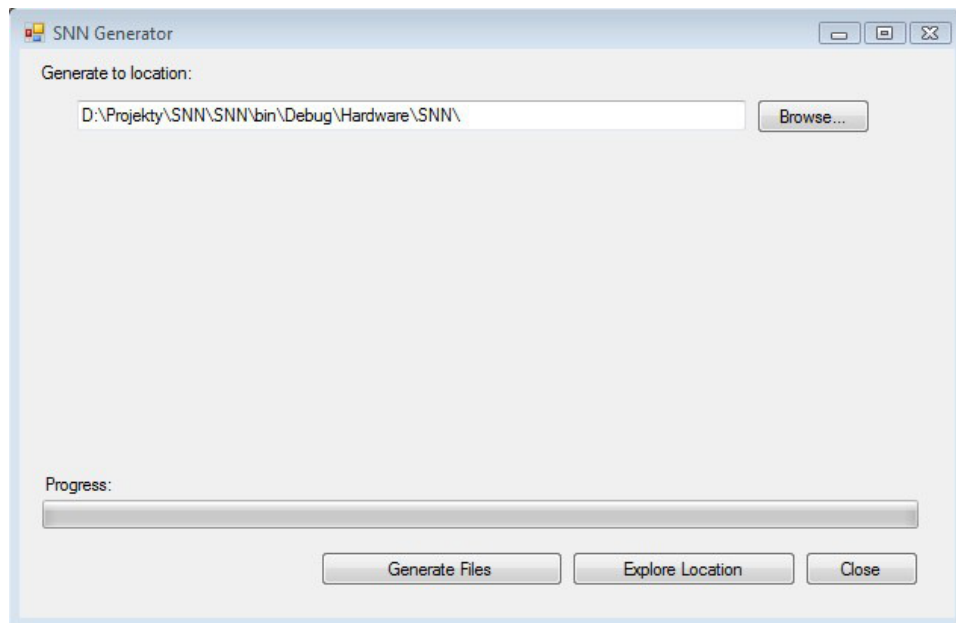
Table 1. Sample VHDL code generator tags

Tag	Description
<PSP_FUNCTION_VALUES>	Encoded values for postsynaptic potential function.
<ZERO_VALUE>	Value for resting potential.
<FREQUENCY_DIV>	Value used by frequency divider to generate 1ms clock(integer)
<NUMBER_OF_BITS>	Length of vectors used inside the hardware system. (integer)

<INPUT_NUMBER>	Number of input neurons(integer)
<LAYER1_NUMBER>	Number of neurons in layer 1(integer)
<OUTPUT_NUMBER>	Number of output neurons(integer)
<PSP_LENGTH>	Length of postsynaptic potential duration in milliseconds(integer)
<INPUT_DELAYS>	Delays of input neurons.
<INPUT_TO_LAYER1_DELAYS>	Delays of synapses between the input layer and first hidden layer
<LAYER1_TO_OUTPUT_DELAYS>	Delays of synapses between the first hidden layer and output neurons
<LAYER1_WEIGHTS>	Weights of synapses between the input layer and first hidden layer
<OUTPUT_WEIGHTS>	Weights of synapses between the first hidden layer and output neurons
<THRESHOLD>	An encoded threshold value.

Every tag should be placed in a new line and it is essential for the file with tags to have a format:

<tag>=value



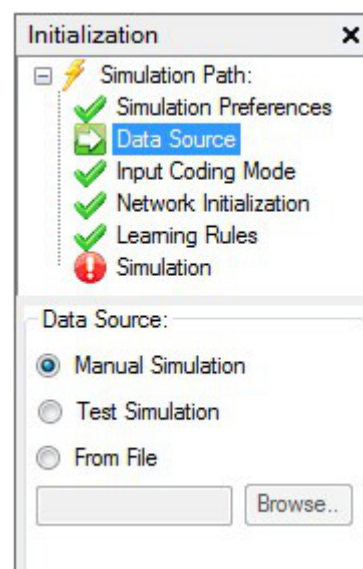
**Figure 10. VHDL code generator window.**

### 3.6. Simulator

When user selects an *Initialize Simulation* from the *Simulation* menu a window shown in Figure 11 appears. It is used to guide user through simulation initialization process. On the top, one can see *TreeView* component that represents simulation path. It is obligatory to go through every step of this path and currently taking step is indicated by green icon with arrow, application validates option chosen by user and if everything is correct - green 'OK icon' appears while if

something has been chosen wrong or some step has been omitted red icon with exclamation mark appears. Following simulation steps has been designed:

- Simulation preferences – allows user to select basic simulation options
- Data source – allows selecting the source of stimuli. A source can be the text file or manual stimulation, also an additional ‘test’ option has been added to run network without any particular task.
- Input coding mode – allows to select way of stimuli coding, if an Image Recognition option has been selected, also it is necessary to insert input pattern dimensions.
- Network Initialization – in this step user can adjust a weight or delay initialization parameters
- Learning rules – parameters for the STDP algorithm if neural network learns.
- Simulation – is used to carry out the simulation, when a manual simulation mode has been chosen this step provides components and solutions by means of which user can stimulate input neurons.

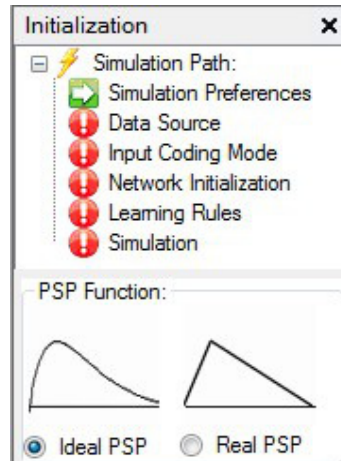


**Figure 11. Simulation initialization window.**

In figure above one can see a simulation initialization process at the fifth step, however user got back to the second step: *Data Source*, which is indicated by arrow. Last step of initialization hasn't been taken yet.

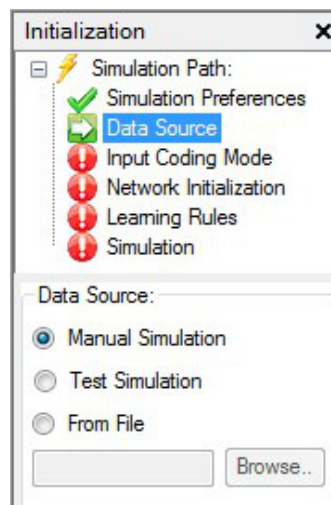
### 3.7. Simulation initialization

This section will describe every step of the simulation initialization process. In first step of simulation initialization user should select if postsynaptic potential function is ideal or its real approximation used in hardware implementation.



**Figure 12. Initialization – simulation preferences step**

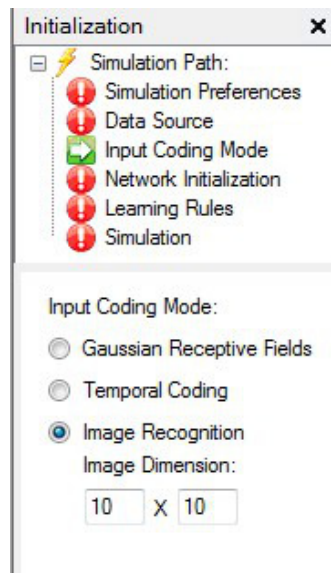
In Data Source step, user can define the source of stimuli. A *manual* simulation option allows to feed the network with manual data input by using simulation panel.



**Figure 13. Initialization – data source step**

The *test* simulation allows running the network in test mode where it does not process any task and input data is no necessary. In this mode input neurons fire spikes according to its delay parameter.

In third step, user can define how input stimulus will be encoded:

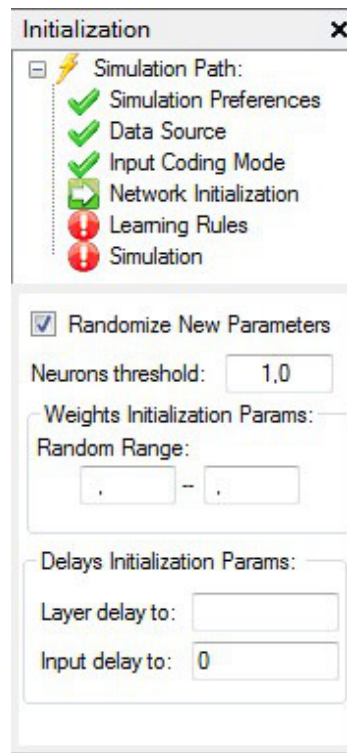


**Figure 14. Initialization – Input Coding Mode step**

Three options are available:

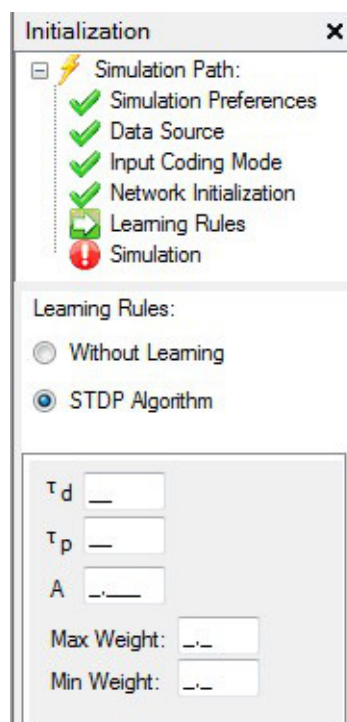
- The Gaussian Receptive Fields - Gaussian Receptive Fields coding.
- Temporal Coding - basic temporal coding method.
- Image Recognition - if this option is selected, pattern dimension in pixels has to be specified.

In the fourth step of the initialization process, the neuron parameters can be randomized if it is desired. If a *Randomize New Parameters* checkbox is not checked all parameters that have been assigned to a network before are used, otherwise user can define the new parameters. Weights and Delays are randomized, while a threshold is fixed according to the value given in *Neurons threshold* box. An acceptable threshold range is from 0.01 to 9.99. To randomize weight of synapses set weight range by fill in Random Range boxes. Possible range is from -9.99 to 9.99.



**Figure 15. Initialization – network initialization step**

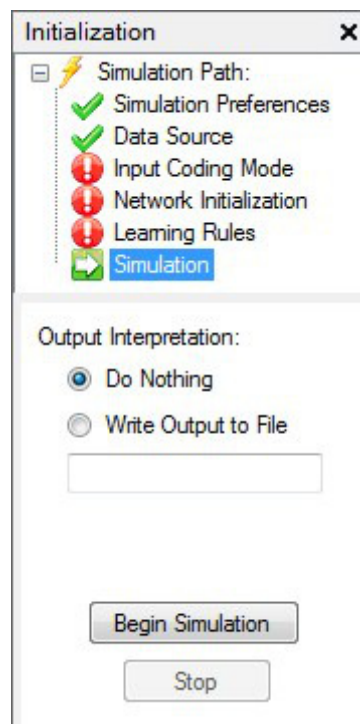
In this step the user can decide if simulation process will train the network by using Spike-Timing Dependent Plasticity algorithm or not.



**Figure 16. Initialization – Learning rules step**

If user has decided to train network, he should also assign STDP algorithm parameters.

In the last step user can begin a simulation process. The network output can also be stored in the text file. The save dialog control will appear after the *Write Output to File* radio box has been selected.



**Figure 17. Initialization – simulation step**

This step also provides two buttons for beginning and stopping the simulation process. After clicking *Begin Simulation* button, if a manual simulation has been selected, a *Manual Simulation Window* will appear (Fig 21) where user can carry out simulation process. The *Stop* button allows to stop simulation every time it is necessary.

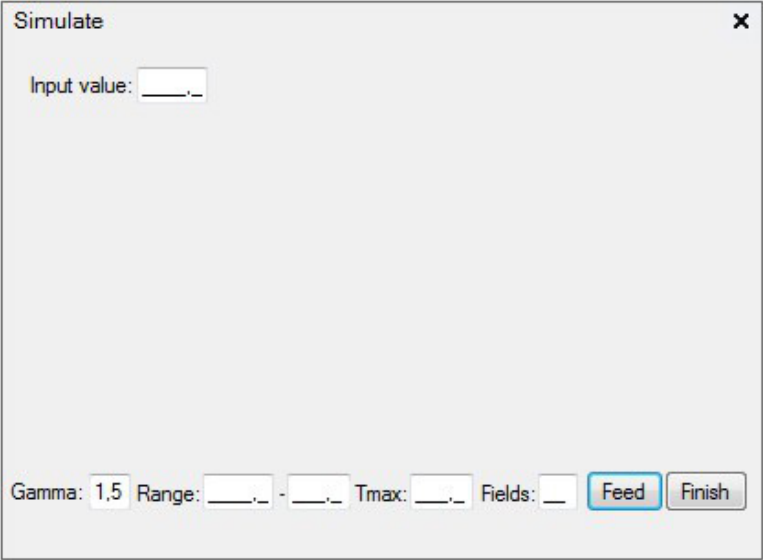
### **3.8. Simulation Process**

In a *Simulation From File* process the network is fed with stimuli from a text file. It is important for this file to hold every stimulus in a new line. During simulation, every fixed time constant a new stimulus is presented to the network. In *Data Source* step of simulation initialization user defines how many epochs is stored within a file and also what is the time of single stimulus presentation. This simulation ends when all stimuli from a file has been presented.

In a *Manual Simulation* mode, user has to use the *Simulation* panel to feed network with stimuli. Dependent on what stimuli coding method chosen the right controls will be shown.

If Gaussian Receptive Fields coding method has been chosen, then following window is shown:





Simulate

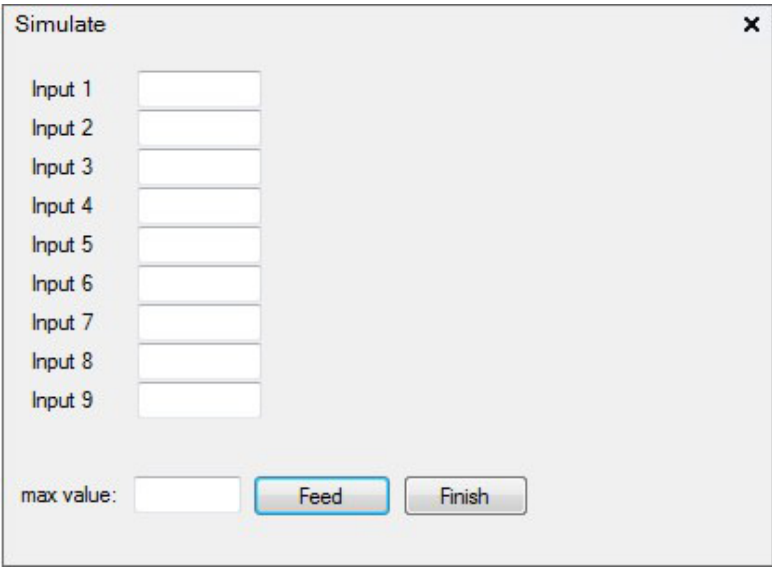
Input value:

Gamma: 1,5 Range:  -  Tmax:  Fields:

**Figure 18. Manual simulation – Gaussian Receptive Fields mode**

There user can set the Gaussian Receptive Fields method parameters. An *Input Value* text box allows to insert value which with the network will be stimulated.

If the *Temporal Coding* method has been chosen a window shown below will appear:



Simulate

Input 1

Input 2

Input 3

Input 4

Input 5

Input 6

Input 7

Input 8

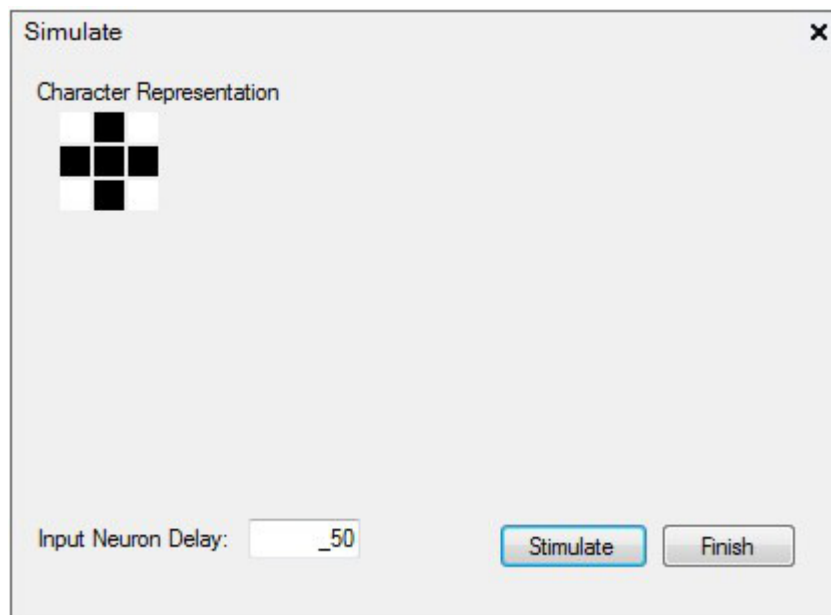
Input 9

max value:

**Figure 19. Manual Simulation – Temporal Coding mode**

Every neuron will be fed with the value inserted in textbox corresponding to it according to Temporal Coding method where time window corresponds to value given in *max value* textbox.

When Character Recognition simulation has been selected, the following window will appear:



**Figure 20. Simulation – Character Recognition Mode**

The character is represented by a matrix of clickable images every of which corresponds to the input neuron. If the square is black it means that input neuron will fire a spike, otherwise no spike will be fired. Squares can be clicked during simulation process without stopping it. The *Input Neuron Delay* textbox allows to set a time period which will characterize the period of input neurons firing time. This value should not be very small.

### **3.9. Application Options**

Application Options form allows to change various parameters used by software, including its appearance, and functionality. *Treeview* on the left side of form makes easier selection between options sections:

- Environment
  - General – Allows to modify basic options
  - Colours – Allows to modify appearance and colours used in application, mostly by graphs.
  - Paths – Allows to modify paths of application directories
  - Others – other environment parameters and options
- Simulation
  - Neuron Behaviour – to set some neuron parameters

- Graphs and plotting – to set behaviour of plotters during simulation
- VHDL code generator – basic options for the file generator
  - Other options – to specify more advanced options used during file generation.

### 3.10. Application Help

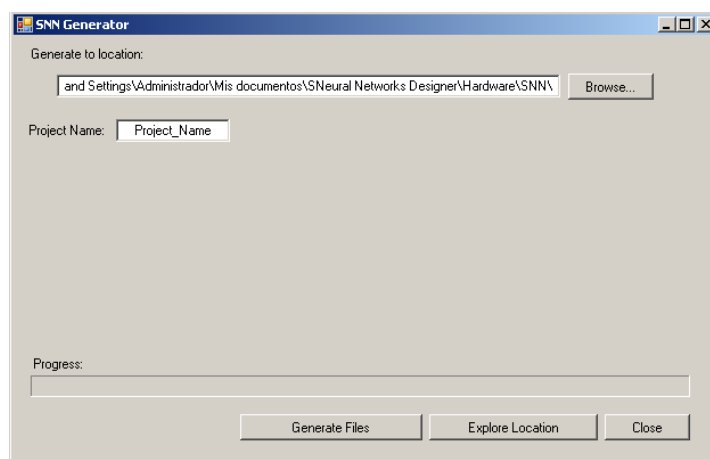
To overview the application options, modules and functions a Help file has been provided. By using Help file user can learn about the application modules and how to use them. Help content is divided into following sections:

- General Information – introduces basic information about software including the installation requirements and licence
- Introduction – introduces few basic options
- Simulation – provides help for the simulation process
- Tools Menu – introduces and overviews in detail all software tools

Help also allows to search through keywords to find the desirable information.

## 4. VHDL code generation usage.

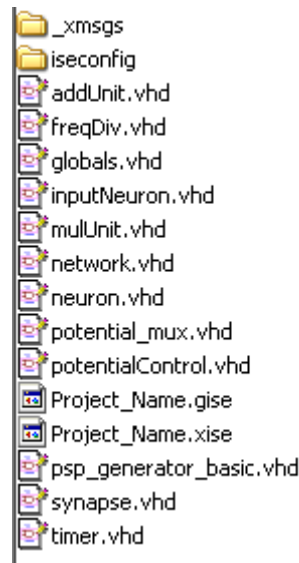
To generate VHDL code for created network, “Tools->VHDL code generator” menu entry is used. After selecting it, a window shown on Fig. 23 appears.■



**Figure 21. VHDL code generator.**

User should select the desired location of the generated code and project name. After pressing the button “Generate Files” the code is generated and placed in the given location.

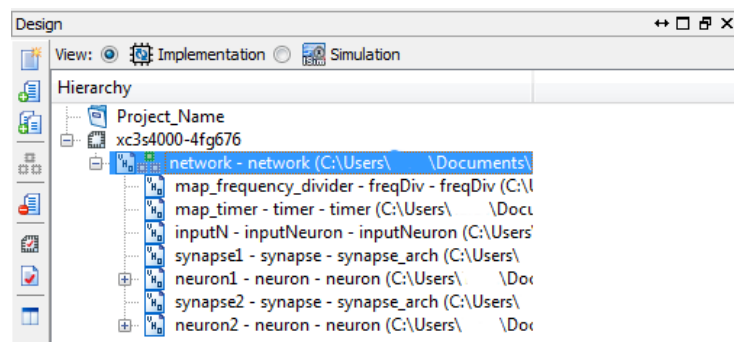
The list of files generated is shown on Fig.24.



**Figure 24. Files generated.**

The generated files include project files specific for Xilinx ISE versions 8.1-9.2, but the VHDL code can be used in any HDL IDE with manual import. In case of Xilinx ISE, the project can be directly opened from the location. If newer versions of ISE are used, the project should be migrated to the latest version (Xilinx ISE automatically suggests this action).

After opening the project in Xilinx ISE, the network is shown as prepared instance, ready for simulation or connection to other parts of the design (Fig. 25).



**Figure 25. The project successfully opened in Xilinx ISE 14.2. “Network” top instance contains input and output ports ready to be connected to other elements of the final system.**