4 Schematic Capture System

The Schematic Capture System (SCS) provides graphical tools to build hierarchical neural models either by a top-down or bottom-up approach. SCS consists of the Schematic Editor, Icon Editor, NSLM Editor, Library Path Editor, Consistency Checker, Library Manager, NSLM Code Generator, and NSLM Viewer. SCS allows one to build a model graphically by connecting icons together into what we call a schematic. Each icon can then be decomposed further into a schematic of its own. In addition, SCS also provides an interface to the USC Brain Project, Brain Models on the Web database (BMW)

The Schematic Capture System (SCS) is an important component of the NSL system. SCS is primarily used to generate NSL models as shown in figure 4.1.[1] (This chapter covers the latest version of the software found in NSL3_0_n., database version 4.)

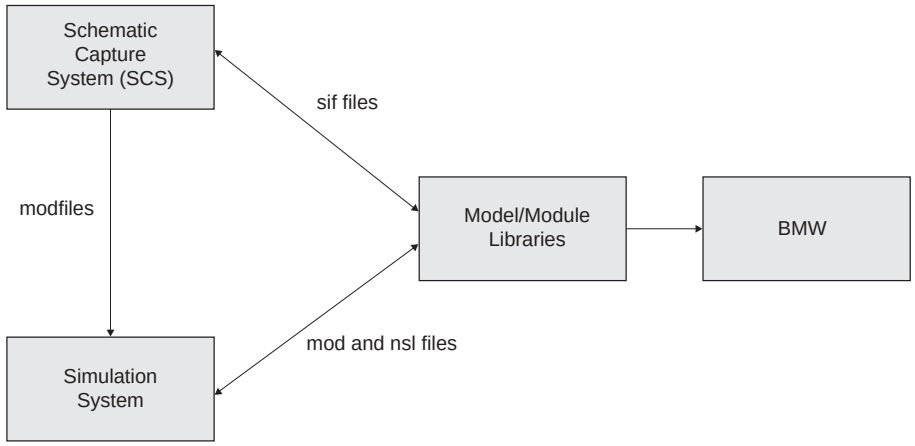

Figure 4.1
NSL System Diagram.

4.1 SCS Tools

The Schematic Capture System consists of many subsystems—the Schematic Editor, the Icon Editor, the NSLM Editor, the Library Path Editor, the Library Manager, the Consistency Checker, the NSLM Generator, and the NSLM Viewer.

Schematic Editor (SE)

The Schematic Editor is responsible for building the structure of the model. It is also serves as the control window for the Schematic Capture System. From the Schematic Editor window we can start any of the other SCS tools, load a model or module into the Schematic Editor, or descend/ascend into a schematic. When selecting icons to use in the schematic, SE allows the user to pick which version of a module to use: the user can choose a floating version that can change at any time or a fixed version, which cannot change.

When opening other tools from the Schematic Editor it is import to note that each new tool pops up in its own window and we can have as many open as we would like. However, there is always one and only one Schematic Editor Window open at any time.

Icon Editor (IE)

The Icon Editor allows the user to build the graphical appearance of the individual icons (modules).

NSLM Editor (NE)
The NSLM Editor allows the user to add NSLM code to the code that SCS has generated. This is particularly important for "leaf" level modules since they contain most of the functionality of the module.

Library Path Editor (LPE)
The Library Path Editor allows the user to modify the list of libraries in use.

Library Manager (LM)
The Library Manager allows the user to access and create new libraries of models and modules within the file system, move module from one library to another, and edit module attributes.

Consistency Checker (CC)
The Consistency Checker is responsible for keeping track of the versions of the modules that the model contains and checking that the ports from one level match those of the next level. The Consistency Checker is called automatically when a model is generated (NSLM Generator) or when a module is saved using Schematic, Icon, or NSLM editors.

NSLM Generator (NG)
The NSLM Generator generates the code from the schematic structure of the model. It also calls the Consistency Checker

NSLM Viewer (NV)
The NSLM Viewer displays the code generated by the NSLM Generator.

4.2 An Example Using SCS

We start by invoking the *Schematic Capture System* with scs.

The *Schematic Editor* window is shown in figure 4.2.

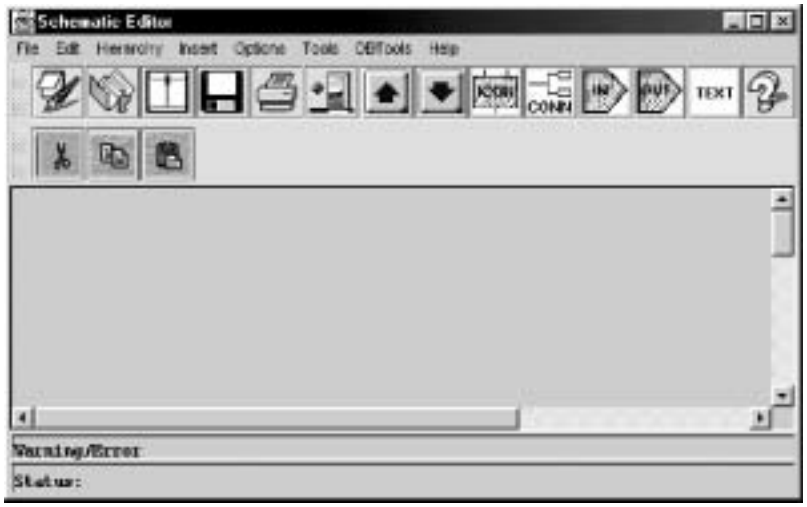

Figure 4.2
The Schematic Editor Window.

There are a number of steps to follow in creating a new schematic: (1) Create a library to save your work in; (2) Create the icons or borrow existing ones; (3) Place the icons in the schematics(4) Connect the icons together; (5) Save the schematic back to one of the libraries; and (6) Generate the NSLM file.

Create a Library
From the Schematic Editor window choose the Tools menu and then select the Library Manager option. (We will abbreviate this to: "**Tools→Library Manager**" in the future.) The system opens the window shown in figure 4.3.

Figure 4.3
The SCS Library Manager Window

Verify that the first library is "<*somepath*>/nsl3_0/BookLib" (the "/" directory symbol in UNIX corresponds to a "\" symbol in a PC) where *somepath* is where your administrator installed the basic SCS library. Create another library in which to save your schematics by selecting **Library→New Library**. A popup will appear as shown in figure 4.4.

Figure 4.4
New Library Path Prompy

Enter the path where you wish to create your library, **in our case it is f:/usc/ns/NSL3_0_m/nsl3_0/FirstLib**. Then select **OK.** When you are finished, select **Close** from the Library Management Window.

Create Icons

To create a schematic, we first need to verify that the icons we want exist. In this example, we will start from scratch and create icons for the **Ulayer** and **Vlayer** modules we wrote earlier. First open the *Icon Editor* where individual icons/modules are edited. This is achieved by **Tools→IconEditor** from the Schematic Editor Window. A pop-up window will appear as shown in figure 4.5.

Figure 4.5
The picture shows the layout of the Icon Editor Window. The top row of menu options allows us to create new icons, edit old ones, and change the graphical options. The left tool bar is in charge of the graphical editing commands.

Since we want to create a new icon we select **File→New** from the top menu bar. In response to this option we get the window shown in figure 4.6.

Figure 4.6
New Icon Prompt

In figure 4.6 we note that the first thing in the Icon Prompt Window is the library name. By default the last library we enter is the first library on the list. Next, we type in the name **Ulayer as the icon or module name (the icon is just one view of the module).** We also note here that the first letter of **Ulayer** is capitalized since it is a module and not an instance of one. Next we specify the version number of the module or icon we are creating. We will take the default 1_1_1. Next we specify the icon type corresponding to the type of template that we want to specify. At this point we choose **NslModule** since we are about to specify a module (see table 4.1), and we would like the buffering to be "false" for non-double buffering (see table 4.2). Next we select the option of "float all submodules" which allows specify the default option to apply to submodule of this module (see table 4.3). This will be explained in more detail later. Finally, we specify the arguments for this module/icon, and there is only one "int size".

Module Types	Description
NslModule	leaf and middle level modules
NslModel	top level module
NslClass	user defined class
NslInModule	stimuli
NslOutModule	output displays

Table 4.1
Module Types

Buffering Choices	Description
true	double buffering of output ports—this option is for simulated parallel processing
false	no buffering—this option is for sequential processing (default)

Table 4.2
Buffering Choices

Get Newest Version of Submodules	Description
true	Specify a default that submodule versions may change
false	Specify a default that submodule versions may not change.

Table 4.3
"Get Newest Version of Submodules" Choices

When we are finished we select "OK". You should see a figure similar to that shown in figure 4.7.

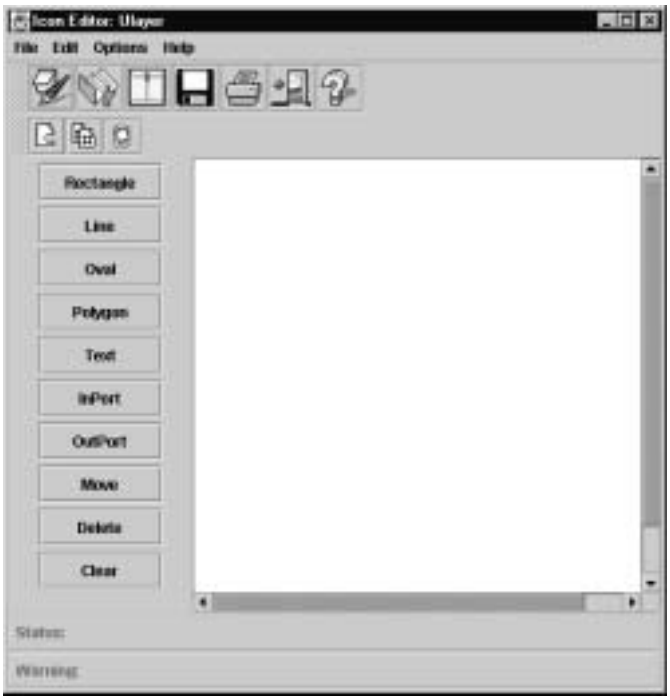

Figure 4.7
Icon Editor Window after Ulayer Module just created.

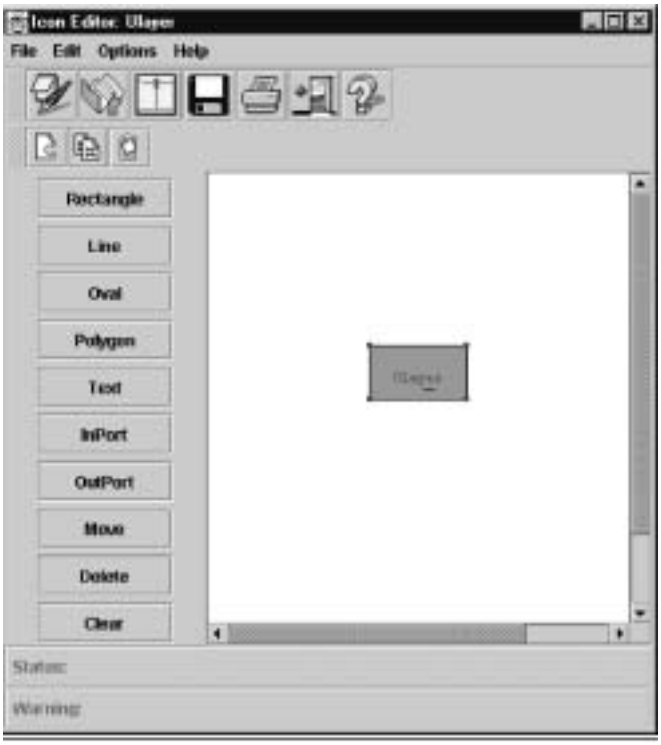

Figure 4.8
The Ulayer's Icon without Ports.

We then go back to the Icon Editor Window where we press the **rectangle** button. To get the rectangle icon in the canvas we first move the mouse to the canvas window and then drag the mouse across the screen until the rectangle is the desired size. The output of this is shown in figure 4.8.

Specifying the Ports on the Icon

Now we want to add two input ports, *v_in and s_in*, and one output port, *uf*, to the icon. This is done by selecting "InPort" and then selecting "OutPort". A popup window will appear similar to the one in figure 4.9.

In this window we first type the name, *s_in*. Then a window which looks like that in figure 4.10 will appear. We specify what kind of data structure the port will hold, mainly **NslDinInt**, **NslDinFloat** or **NslDinDoubl*e***. In this case we choose NslDinDouble. Next we specify the Dimension X where ***X*** represents the dimension: 0, 1, 2, 3, 4, or higher-Dim. Nsl currently only handles dimension of 4 or less but you can create your own user defined type with more than 4 dimensions. In this example, we choose "1" as the dimension. Direction indicates the direction the user would like the port to point "left→right", "right→left", "up→down", or "down→up". We choose "left→right". The "Signal Type" indicates whether the port has an excitatory or inhibitory affect on the module. We choose the signal type to be "excitatory", and the parameters to be just the "size" of the array used. The parameters correspond to the same parameters we would provide in the NSLM language. (*s_in* has 10 elements which will be defined through the "size" parameter.) When done entering, select "**OK**" from the bottom of the window. See figure 4.10.

Figure 4.9
Input Port Name s_in on the Ulayer module.

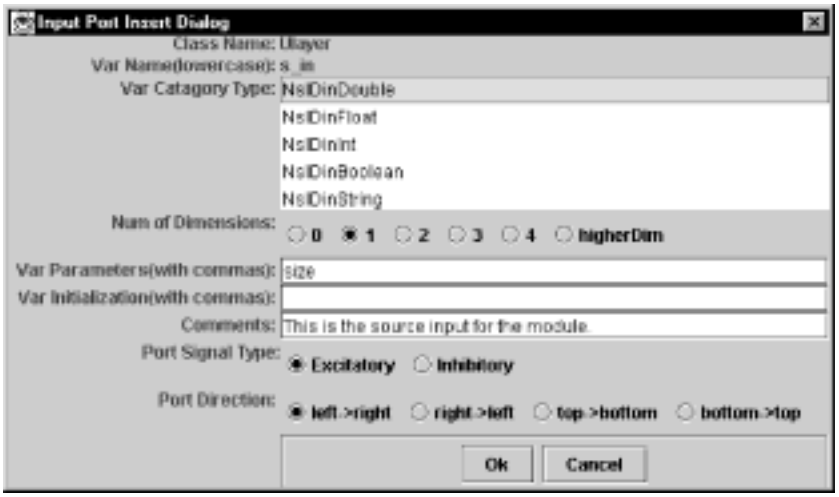

Figure 4.10
Input Port information for s_in on the Ulayer module

Once entered, you will need to specify the position of the pin or port. For convenience, select any spot on the Icon Canvas where you would like the end point of the pin to go. We have selected a location such that it looks like the input is going into the rectangle. See figure 4.11.

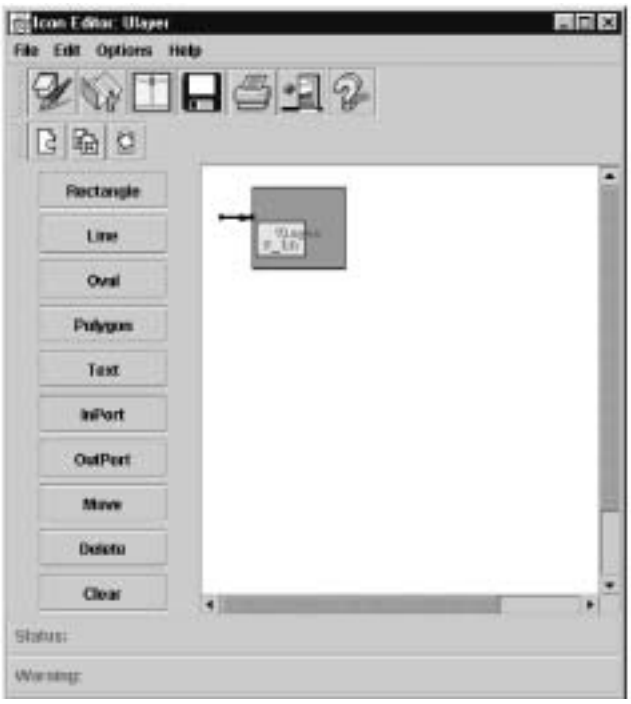

Figure 4.11
Input Port s_in on the Ulayer module

For input port *v_in*, we choose type NslDinDouble0, and we choose "left→right" as the direction. We choose the "excitatory" signal type, and there are no parameters. When done entering, select "**OK**" from the bottom. See figure 4.12.

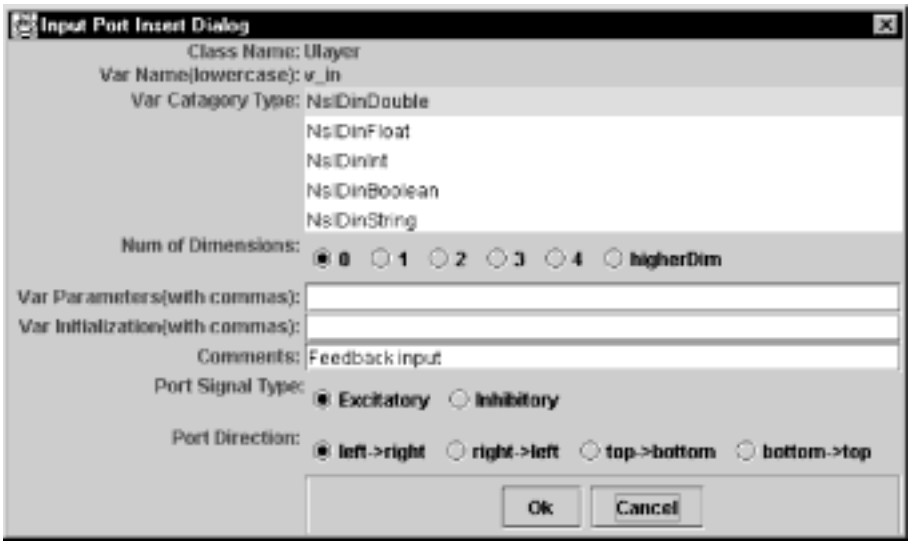

Figure 4.12
Input Port Information for *v_in*

The resulting port is shown in figure 4.13.

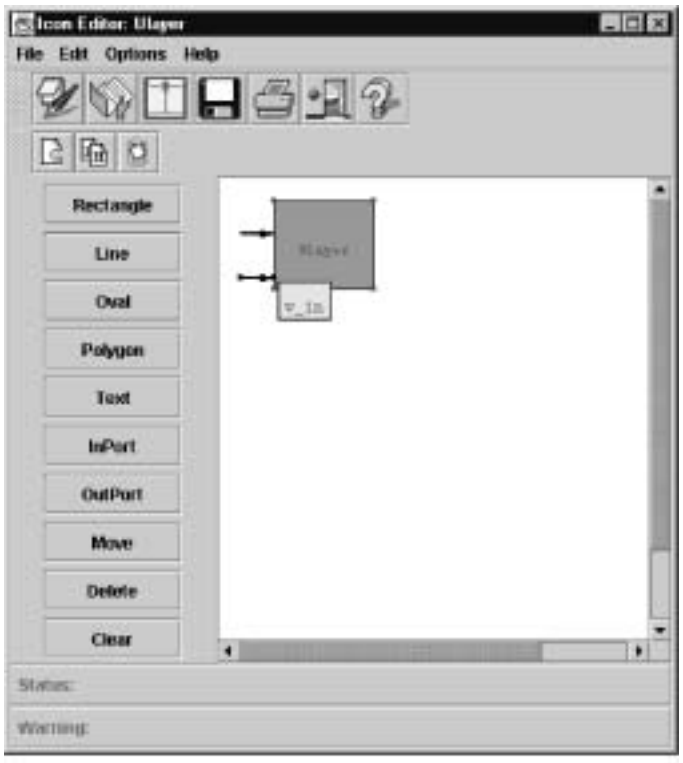

Figure 4.13
Input Port picture for *v_in.*

For port *uf*, we select the "OutPort" button, and type the name *uf*, choose the option "NslDoutDouble1", with direction "left→right", and parameter "size". When done entering select "OK" from the bottom as shown in figure 4.14.

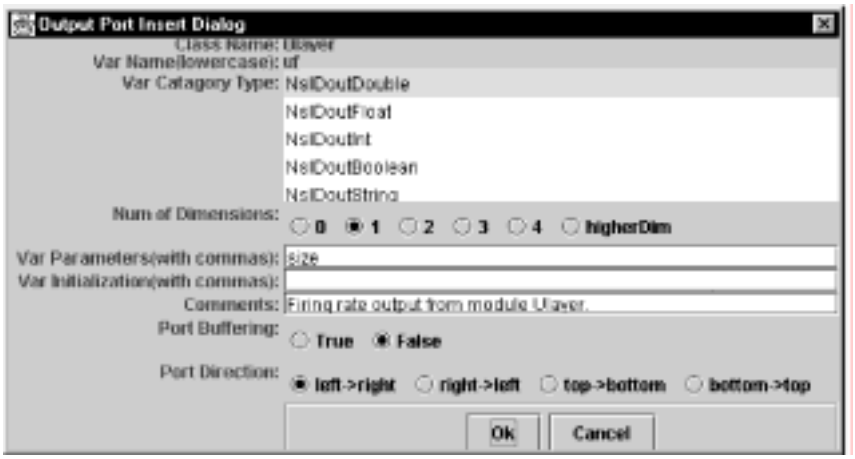

Figure 4.14
Output Port Information for *uf.*

We next save the icon by selecting the **File→Save** menu option from the Icon Editor window. We have now completed the icon creation process and should have an icon with port entries as shown in figure 4.15.

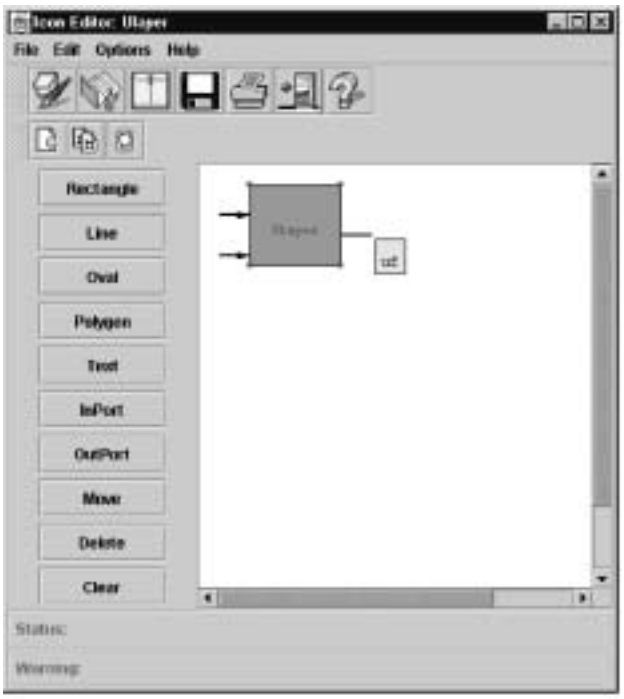

Figure 4.15
Output Port picture for uf.

Now it is time to create the second icon, **Vlayer**. **Vlayer** is created in exactly the same manner as **Ulayer**, except that its input port is *u_in* (with dimension 1 and parameter *"size" without the quotes*) and its output port is *vf* (with no dimension). After completing it, we are ready to move on to creating the schematic of the **MaxSelector** module itself.

Creating the Schematic

To create a schematic from the Schematic Editor Window select the **Module→New Module** menu option. A window should appear similar to the one below. Type in the name "MaxSelector" and version number "1_1_1". Specify the library as the **c:\users\me\nsl3_0\FirstLib** or whatever library you are using. Since this module is going to be a middle level module, we will declare it to be of type "NslModule". When you are finished select "OK" as shown in figure 4.16.

Figure 4.16
New Module Creation Window for MaxSelector.

Now we are going to add the two icons we just created and then connect them. First select the **Insert→Icon** menu option. A popup window will appear, similar to figure 4.17. We type in the instance name of u1.

Figure 4.17
Submodule instance name popup dialog.

Next, a popup window similar to the one in figure 4.18 appears. We fill in the instance information: which is the instance name and the instance parameters. In this case, u1 and size. Instead of typing in the name of the library, module, and version, we simple select the "Or Choose File" option and select the **Ulayer** icon from the library as shown in figure 4.19.

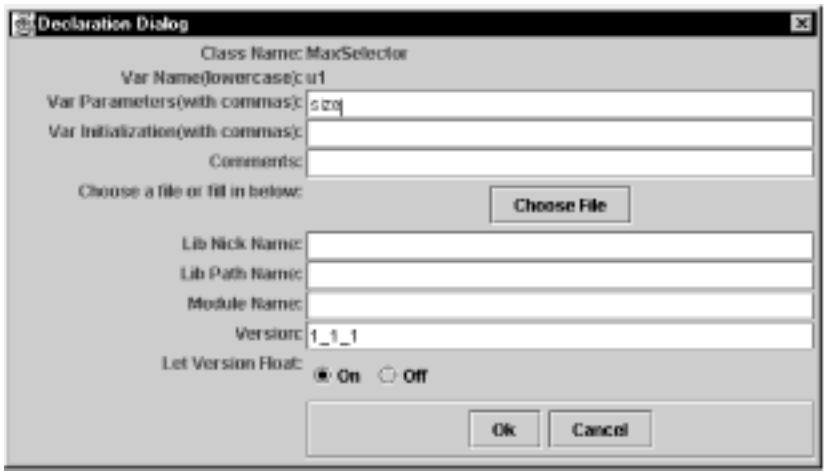

Figure 4.18
Choose submodule or icon popup window.

In figure 4.19 if we click on the library name we want (in this case the first line), then we will see a list of modules to choose from. If we click on the **Ulayer** module, we will see the different versions of this module as shown. For this exercise, we will version "1_1_1".

Figure 4.19
Selection of the Ulayer module from the Declaration Dialog "Choose File" popup.

Finally, we return to the Declaration Dialog box, and the fields for library, module and version are filled in for us as show in figure 4.20. The "Let Version Float" option allows us to specify that we want to take the most recent version of the module or icon—always. This means that even if someone else changes a submodule, we want the latest updates. If we do not want the changes to the submodule, say **Ulayer,** to affect our schematic, then we should set the option to "Let Version Float" to false. Finally, we select "OK" as shown in figure 4.20.

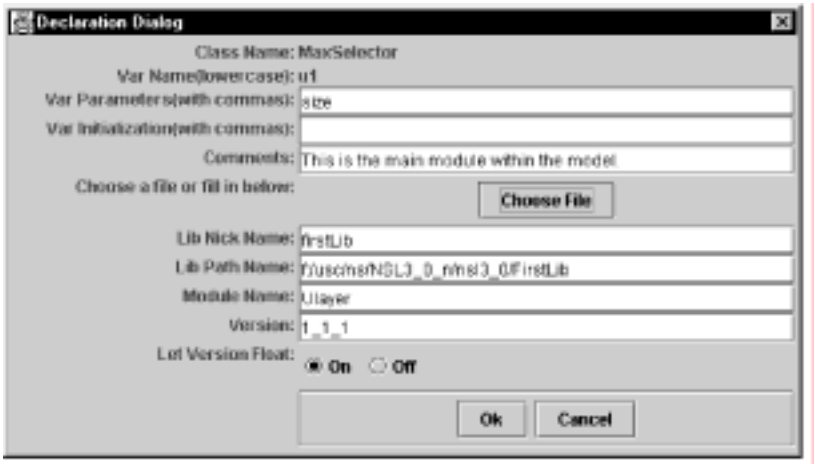

Figure 4.20
Filled in Open Module Dialog Box.

The **Ulayer** icon with an instance name of **u1** will appear on the Schematic Canvas. You will need to take your mouse and select the icon and move it to where you would like it to be located. See where we put it in figure 4.21.

Figure 4.21
Icon placed on schematic.

Next select the "Insert→Icon" command, and use the **Choose File** button to find the **Vlayer** icon template name we just created. Give it an instance name of **v1**. And again, you will need to move *v1* to where you would like it to be located. See figure 4.22.

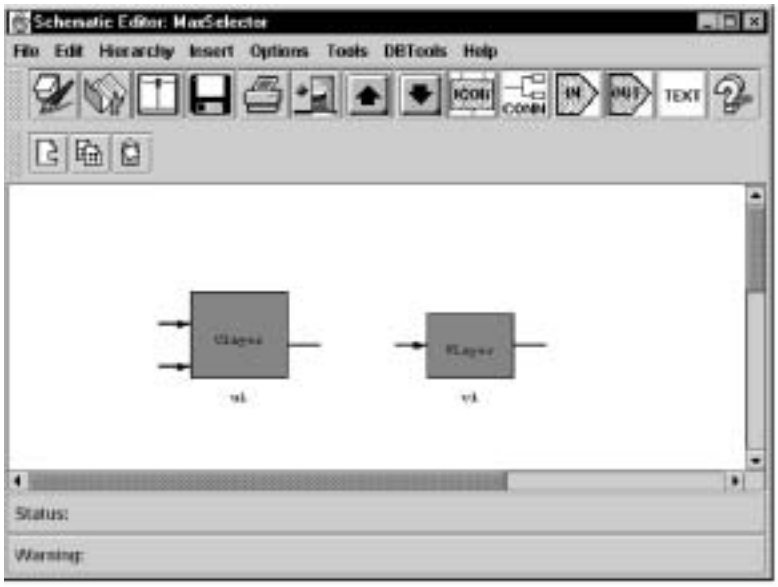

Figure 4.22
The u1 instance and v1 instance placed in a schematic.

Next we must add input and output ports to the MaxSelector schematic. We select the "Insert→Inport" menu option and a pop-up window appears. The name is "in", the type is "NslDinDouble1", the direction is "left→right", the signal type is excitatory, and the parameter is *size*. See figure 4.23.

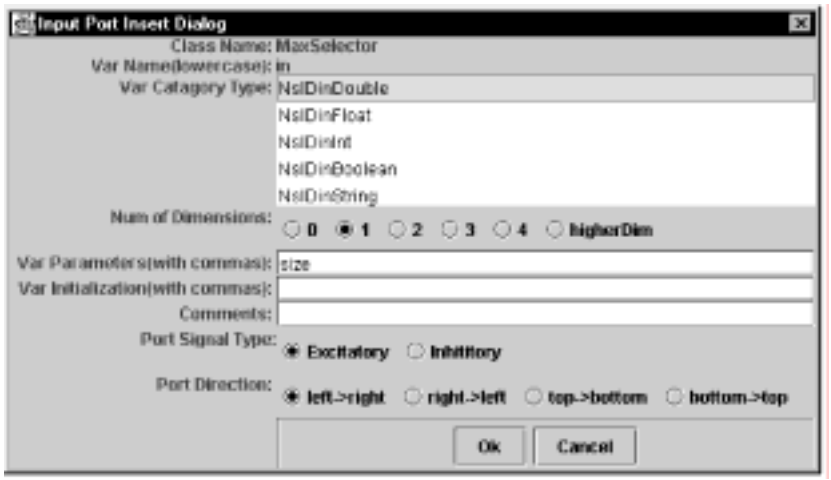

Figure 4.23
MaxSelector input port specification.

Again, move the input port icon into position as shown in figure 4.24. (We sometimes call these ports "inports".)

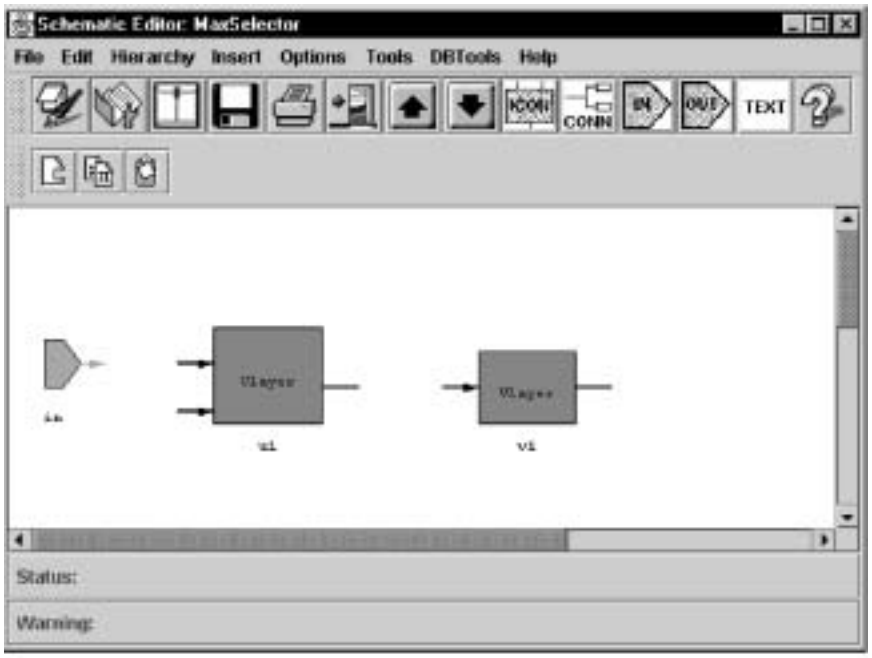

Figure 4.24
MaxSelector input port specification.

We also must add an output port to this schematic. Add an output port by selecting "Insert→Outport" from the menu. The name of this port should be "out", the type is "NslDoutDouble0", the direction is "left-→right", the buffering is set to true, and the parameter is "size". See figure 4.25.

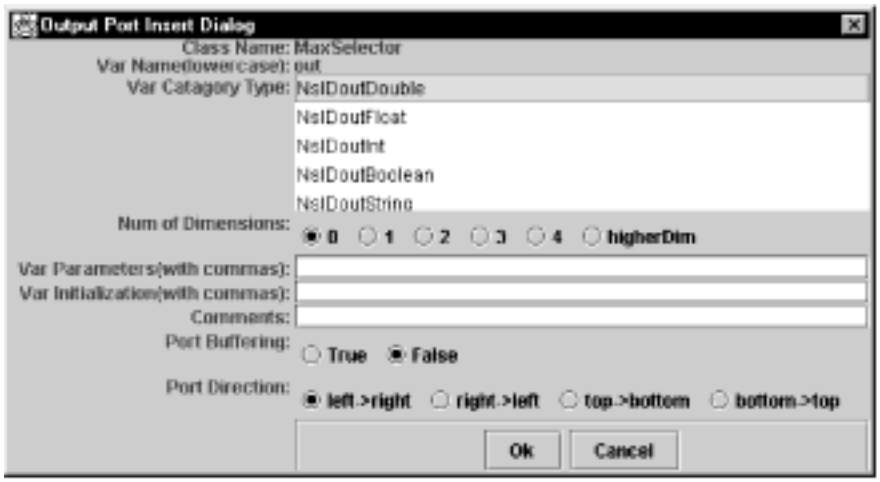

Figure 4.25
MaxSelector output port specification.

You should now see the picture in figure 4.26.

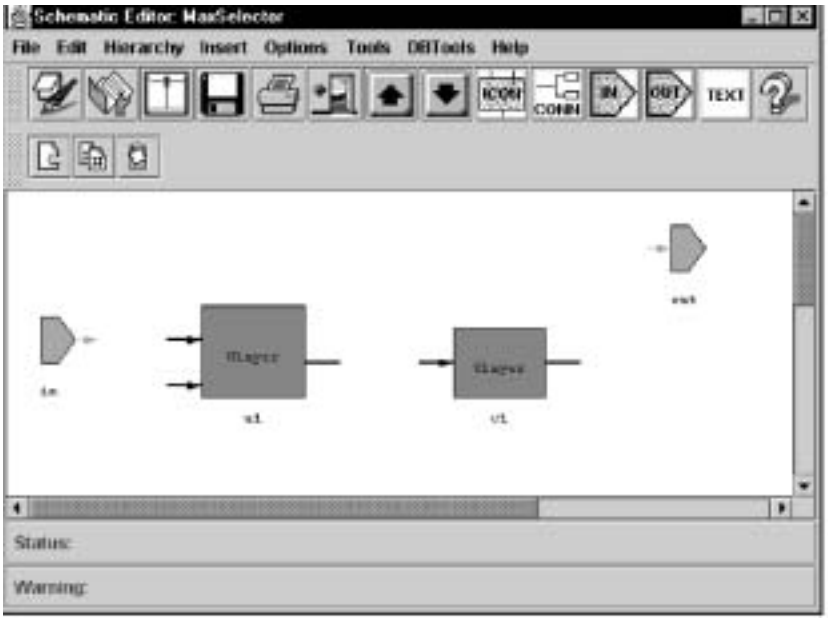

Figure 4.26
All icons placed in the MaxSelector module.

Finally, with all of the icons in place we are ready to add the "interconnect". Select **insert→connection** from the Schematic Editor menu. (But before doing so make sure you do not have anything else selected. You can unselect an object by click on the right mouse button.) You can tell that you are in "connection" mode by the status window at the bottom. It should say "Insert Connection". Let us connect the icons moving from left to right. First, place your mouse over the output pin of the input port "in". Push the mouse button down. Next, drag the mouse to the upper input pin on the "u1" icon. Release the mouse about in the middle of the pin. You should see a picture similar to figure 4.27.

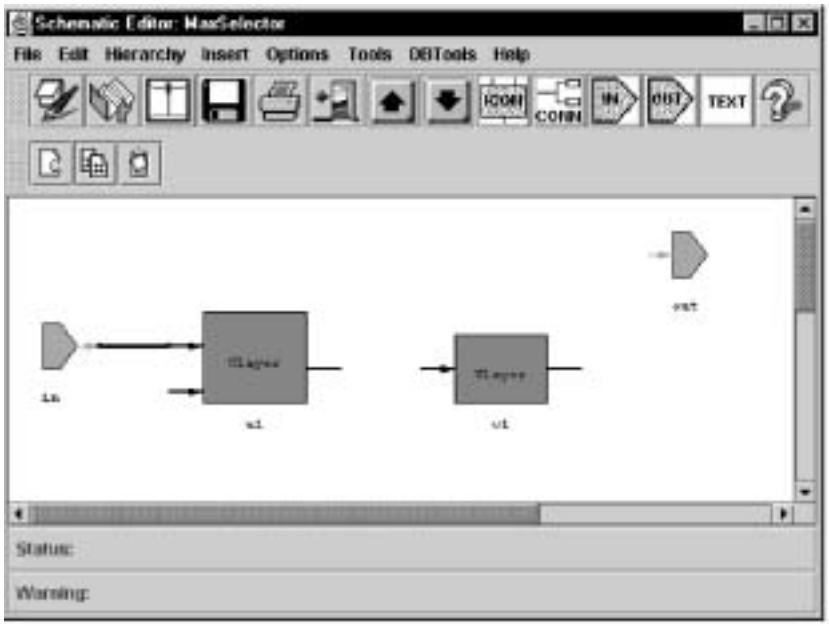

Figure 4.27
First line of interconnect between "in" and "u1".

Next, place your mouse over the output port of the "u1" instance. (First, a little flag with the name of the pin should appear.) Push the mouse down and drag the mouse to the inport of the v1 instance and release the mouse button. Next, place your mouse over the outport of the v1 instance, push the mouse down, and drag the mouse to the right by about one half inch, release the mouse. With the mouse in the same place, press the

mouse down and drag it downward one inch. Release the mouse. With the mouse in the same place, press the mouse down and drag it to the left until it is just past the "u1" input pins. Release the mouse. With the mouse in the same place, press the mouse down and drag it to the lower input pin of the "u1" icon. You should then see the picture in figure 4.28.

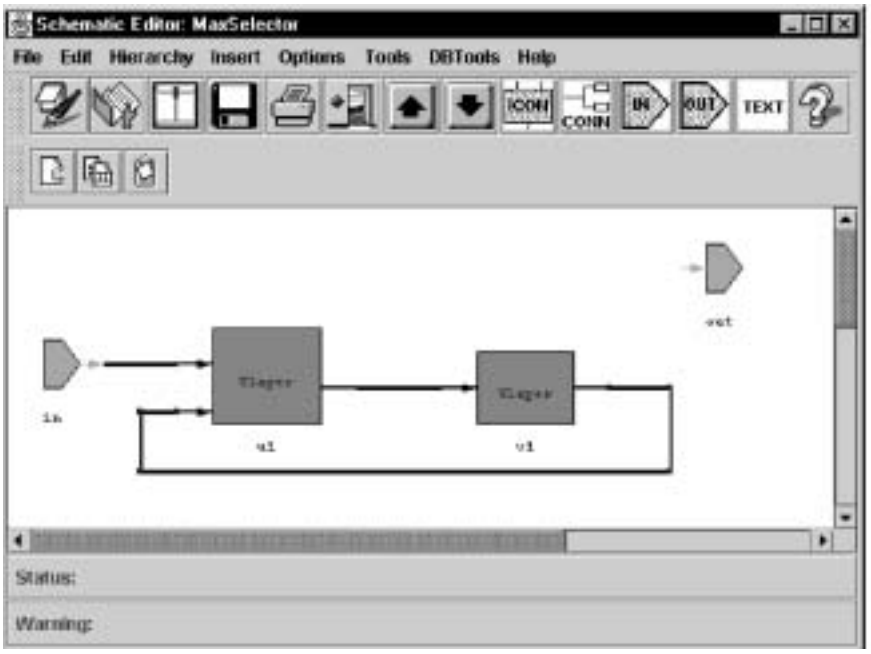

Figure 4.28
Connection between the output port of v1 and the input port of u1.

The last connection we need to make is from the "u1" output port to the output port of the MaxSelector schematic itself. Move the mouse over the output pin of the icon "u1" and push down. Next drag the mouse up about three-quarters of an inch, and release the mouse. With the mouse in the same position, drag it to the input side of the "out" output port icon, and release the mouse.

At this point, we have just completed our first schematic. The result is shown in figure 4.29.

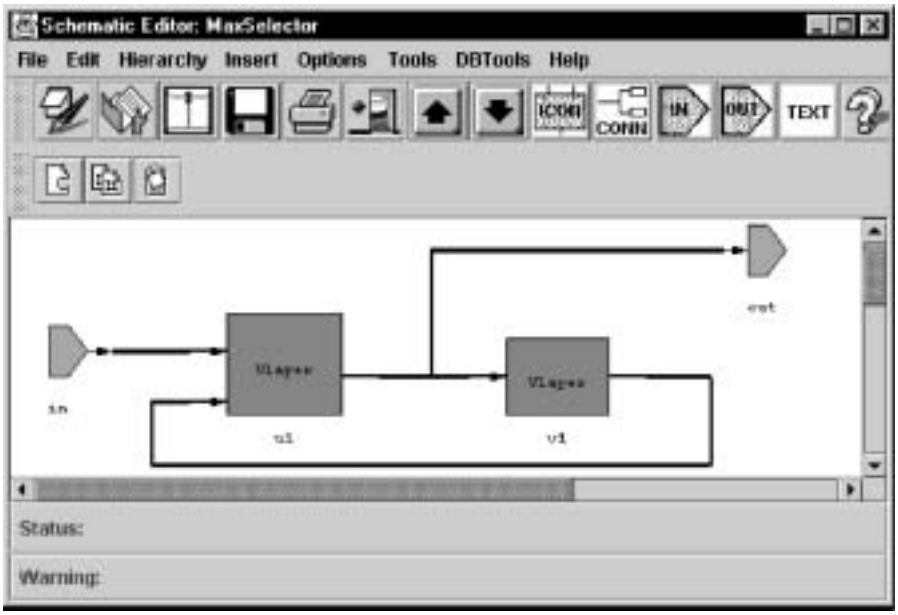

Figure 4.29
Finished Schematic of MaxSelector module.

Mouse Action Commands

Before we move on, we would also like to describe some of the mouse action commands that SCS provides. We have already mentioned the "unselect" option, and here are three more.

- Select one object—The user clicks on any object in the schematic canvas and that object will be highlighted, indicating that it is selected. If the user keeps pressing the Shift key down, then the selected objects will be this newly selected one plus previous selected ones.
- Unselect object—When an object is in a selected mode, then clicking with the right mouse button again will make it unselected.
- Move object—The user can move any object (individual or group object) in the canvas by first clicking on it and then dragging the mouse.
- Descend—In current schematic page, if the user double clicks on a module, then the detailed layer corresponding to that module will be shown in the canvas.

Automatic Generation of Code

After completing the schematic we can see the NSLM code that it generates by selecting **Tools→View NSLM** from the Schematic Editor window. Next, select **File→Open** from the **NSLM Viewer** window and open the MaxSelector module we just created, as shown in figure 4.30.

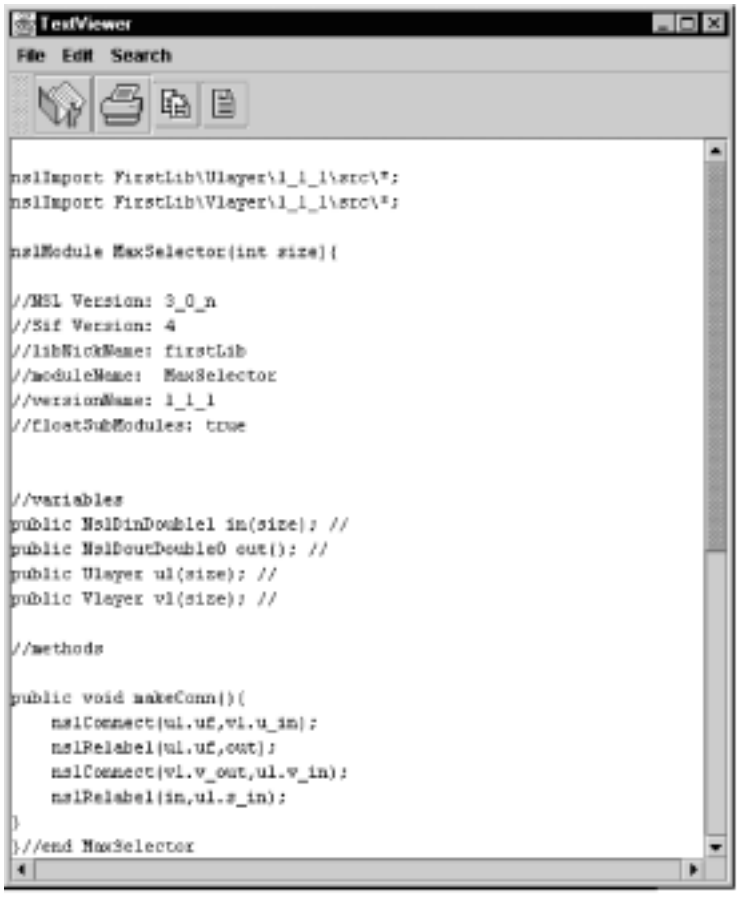

Figure 4.30
NSLM Viewer with MaxSelector module.

We notice that SCS has generated the definition of the module, the ports and the variables for us. It has also generated the "makeConn", but not methods such as "initModule", "initTrain", and "initRun" which are still needed. The we need to fill in these other methods using the NSLM editor. We will examine how to use the NSLM Editor next.

Manual Generation of Leaf Level Code

Although a lot of the code has been automatically generated, we still need to fill in the code for both the **Ulayer** and **Vlayer** modules. We do this with the NSLM editor. Select **Tools→NSLM Editor** from the Schematic Editor window. Next select **File→Open**, from the NSLM Editor window and then select **Ulayer** version "1_1_1" from the list. An editor similar to the following should appear. Notice how **template oriented** this editor is, as shown in figure 4.31.

Figure 4.31
The First Half of the NSLM Editor Window showing the Name, Arguments, and Flags.

Figure 4.32
Second half of the NSLM Editor Window showing the variables and the Methods Editor.

Now we just need to add the internal variables *up*, and *h1*. Add these variables and give them the same data types and parameters as in the figure 4.33 and figure 4.34

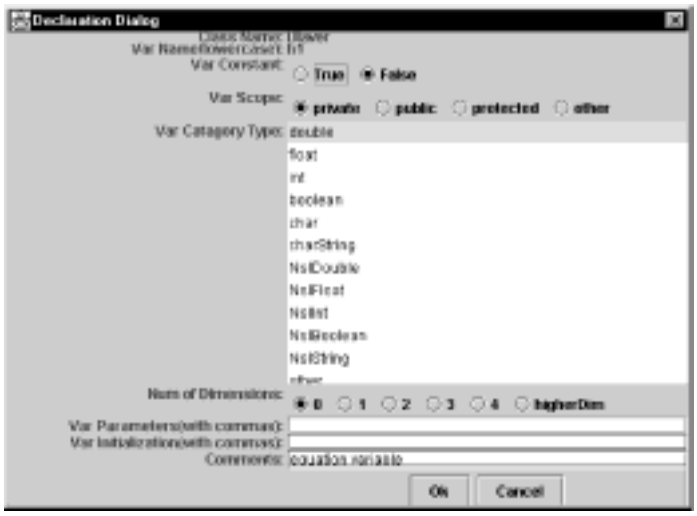

Figure 4.33
Adding the Equation Variable h1 to the Ulayer.

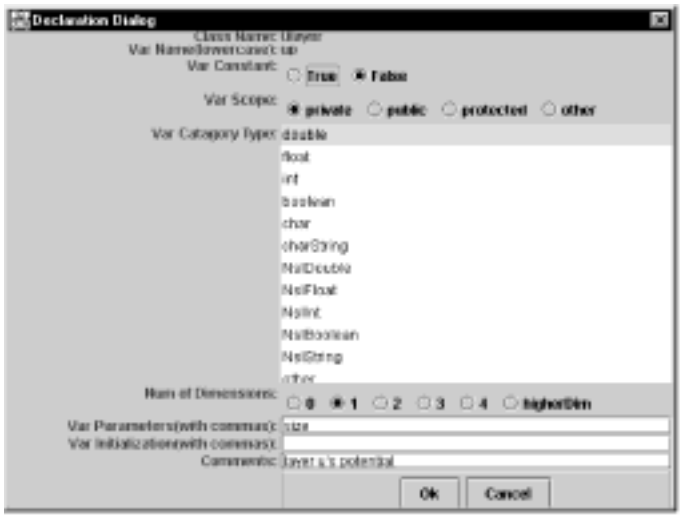

Figure 4.34
Adding Ulayer's potential layer *up*

We note that the output port variables are already declared but not initialized; thus we will initialize them in the initModule and initRun methods within the Methods Window. See figure 4.35.

Figure 4.35
The Ulayer NSLM code.

Next, add the **initRun** method and the **simRun** method as shown in the figure 4.34. And then select **File→Save**. Now do the same for **Vlayer** using the code in figure 4.36.

Figure 4.36
The Vlayer NSLM code.

Generating NSLM Code

We can generate the NSLM code for our modules at any point in the development. For the MaxSelectorModel we need to generate the top level module called the model—MaxSelectorModel (A popup window will appear similar to figure 4.16). We also need to create the modules "MaxSelectorStimulus" and "MaxSelectorOutput", see figure (3.12) and code segements 3.13, 3.14, and 3.15.

To generate the NSLM (mod or module) files for both the NSLJ System and the NSLC System, the Schematic Editor menu select "Tools-→Generate NSLM". The program performs automatic checks to make sure that the icons used in the schematic match what is contained in the NSLM View, it then generates the NSLM code. Once the icons, schematics, and leaf level modules have been created we are ready to make a *Makefile* and executable code.

Compiling and Generating the Executable File

To generate the *Makefile* and executable code, select "Tools→Build Java Version" or select "Tools→Build C++ Version". (When building the executable code, the system checks to see that all of the "mod" files are created and the proper time stamps are on the files. Thus, we can actually skip the "Generate NSLM Code" step if we plan to make an executable file anyway.) We provide both generate options so that we can execute both systems if we desire. Both commands will prompt for the name of the model executable to be built. (This window is the same as that in figure 4.19) The *Makefile* and the executable file will be generated for your particular platform that you are running on and the particular operating system that you are using. These files can be found in the subdirectory "exe" directory below the version directory. Additionally, models can be compiled from a system shell writing 'nsljc model' for Java and 'nslcc model' for C++. For additional compilation and execution details see Appendix V where web site links are specified".

Reusing Modules and Models

To re-use an existing module, simply select it from one of the libraries and include it the schematic for your new module. To reuse a model, you must rename it. If you add ports to an existing model, it them becomes a module, and you must specify the type as such when you go to save your new module.

Copying Existing Modules and Models

As mentioned above we can copy modules, modify them, and give them new names. To copy a module, simply open the existing module in any editor and save it under a new name (you can also save it under the same name but a different version number).

4.3 Summary

We have introduced the different tools available in the Schematic Capture System in helping the user with model creation. In particular, we have shown how to visually build modules and automatically generate code. Some of the tools we covered where the Schematic Editor, the Icon Editor, the NSLM Editor, the Library Path Editor, and the Library Manager. We also showed how to create a library, an icon, and a schematic.[2]

Notes

1. The NSL Schematic Capture System version is based on Sun Microsystem's Java 1.2 programming language and virtual machine. SCS can only be executed as an application and not as an applet since applets put security restrictions on generating output files. We assume that the user has a two-button mouse attached to the computer.
2. Since SCS is one of our newer applications, we encourage the reader to review the latest documentation and technical reports on the NSL web site. See Appendix V for details.