SIM-PL instruction manual

A. v. Inge

University of Amsterdam

Preface

SIM-PL is an educational tool for designing and simulating digital logic circuits. With the capacity to build in a hierarchical manner complex components from elementary components, and to draw bundles of wires with a single mouse drag. SIM-PL can be used (and is used) to design and simulate entire CPUs for educational purposes.

With the simple toolbar interface of the design- and simulation-tool it is simple enough to facilitate learning the most basic and more in-depth concepts related to logic circuits and computer architectures.

SIM-PL is used by students at colleges and universities around the world, ranging from a brief unit on logic in general-education computer science surveys, to computer organization courses, to full-semester courses on computer architecture.

SIM-PL sources are made available under the GNU General Public License.

Features:

- It is free of charge!
- It runs on any machine supporting Java 5 or later.
- There is a web based version too.
- It includes circuit components include inputs and outputs, gates, multiplexers, arithmetic circuits, flip-flops, and RAM memory.
- It is a complete authoring environment.
- The editor:
  - Is based on a simple rubber band drawing principle
  - Editing is possible in a drawing window or in the hierarchical objects list
- The simulator:
  - Contains a block diagram, program editor and time sequence diagram window.
  - The windows are free configurable.

History in a nutshell

In 1994 the first computer organization & architecture course started with a comparable format as the current one. At that time there was no real pressure on finding a proper simulator. It changed when I switched to a different book in 1999. However, back then simulators suitable for teaching were sparse. The question was whether we should build a homemade simulator. While pondering about this question a simulator with a spartan ascii character-based gui was still used.

In 2001 a freshman listening to the name of Wouter Koolen could no longer bear it. He proposed to write a much better simulator. I must admit that I was quite surprised. Especially when he said: "Next week I will have a first version".

In 2011 Wouter Koolen stepped back, not completely, after a decade being the sole developer of SIM-PL and produced during that time a full blown authoring environment. Anybody who made additions to, or build new architectures with SIM-PL has noticed that Wouter is still willing to contribute.

Anthony 'Toto' van Inge

SIM-PL instruction manual is written for a broad audience.
11-08-2016

Cover illustration:
"Tools", a triptych by Carroll Lassiter[1]

[1] Carroll Lassiter
Oil paintings relating to fishing and tools, steel frame mosaic tables and steel fish. I paint romantic portraits of favourite tools, fishing lines and memorabilia. As a painter, I explore a sense of history. My subjects indicate a period of use. Rust on tools, surfaces worn by handling and contact with the earth or water, all add to the story of a given object. Carroll acquired her skills at UNC-CH and Alamance Community College.

Content

1 Introduction

SIM-PL is a triptych with the following “tools”:

- Editor

  It allows, in a very simple and clearly structured manner, to design basic to complex digital circuits. Designing can be done in two modes: in full graphic or text based. During the full design phase between these two modes switching is always possible. A complex design may consist of a number of imported basic or complex designs of which the internal structure is no longer visible at the highest design level.
- Simulator (Executor)

  It allows to perform a simulation with a designed digital circuits. The simulator is capable to provide a design with input data and in steps or series of steps to simulate a design. This tool has extensive test and debug capabilities.
- Compiler

  It allows to compile “code snippets”, with a syntax comparable with “C”, into SIM-PL assembly code executables.

The instruction manual starts with some simple examples to introduce the three tools. First, the editor and the simulator will be addressed by using an already designed NAND gate. Limited functionality will be shown first in order to have a jump-start with the more complex examples.

1.1 Download and unzip

Be sure to download the latest version (ask the TA’s for its location) of the simulator SIM-PL. It will be be in a ZIP format. After unzipping and opening the folder one can see, among other files, several java executables (.jar files) and no explicit install is required. Furthermore, be sure to have the latest Java VM running on your machine.

Also download ‘SIM-PL components’ (ask the TA’s for its location). It will be in a ZIP format also. After unzipping and opening the folder one can see all sorts of simple components (files with .xml extension) like AND, OR, INV and complex components (files with .sim-pl extension).

2 Prologue SIM-PL Editor.

2.1 Starting the editor

By double clicking Editor.jar the SIM-PL editor should open with a window containing various panes (see figure 1). (if not, it is probably related to a Java VM issue)

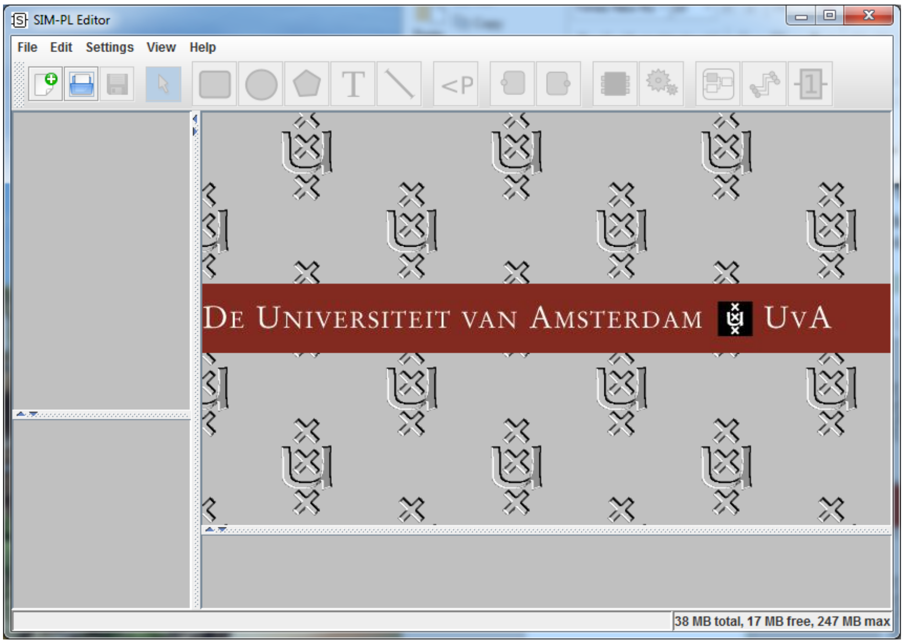

SIM-PL Editor
File
Edit
Settings
View
Help
De Universiteit van Amsterdam
UvA
38 MB total, 17 MB free, 247 MB max

Figure 1. The SIM-PL editor

These panes will be populated when creating a new component or when opening an existing project (see for instance figure 2 or figure 3, for details see 2.5). At the top the menu and just below a toolbar with various buttons; treated hereafter in this instruction manual. The centre pane is the worksheet for editing the schematics. The left part consists of a top and a bottom part. The top part shows the elements in the project in a so called ‘tree view’ pane (see figure 2). The bottom part is a property pane that is used to edit the properties of a specific element in the tree that is selected in the ‘tree view’ pane (see figure 3).

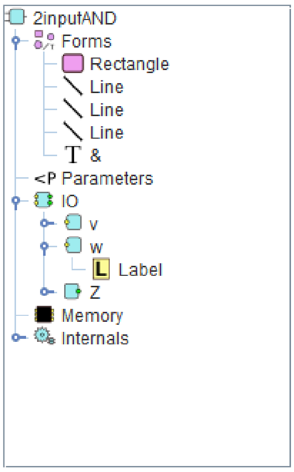
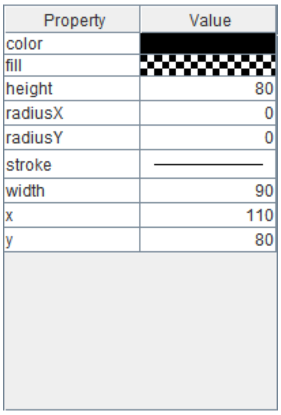


Figure 2. Tree view pane showing the elements within the project



Property	Value
color	■
fill	▨
height	80
radiusX	0
radiusY	0
stroke	—
width	90
x	110
y	80

Figure 3. Property pane with element information

- Tree view pane
  - Forms
    lists the graphic description of a component. (for more details see 2.5)
  - IO
    lists the component labels of the inputs and outputs. (for more details see 2.5)
  - Memory
    See part 2.
  - Internals
    lists the component function. (for more details see 2.5)

2.2 Creating a new component

To create a new component project click on the button, by pressing Ctrl + n or by using the menu File →New. It opens a dialogue window (see figure 4) where one chooses whether the component should be a simple or a complex component. Finalizing by giving the simple or complex component a name (component names can always be changed afterwards).

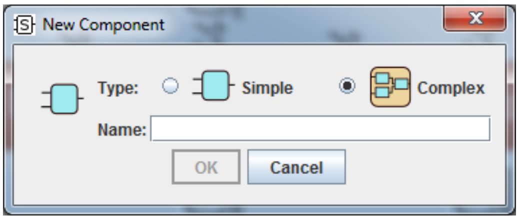
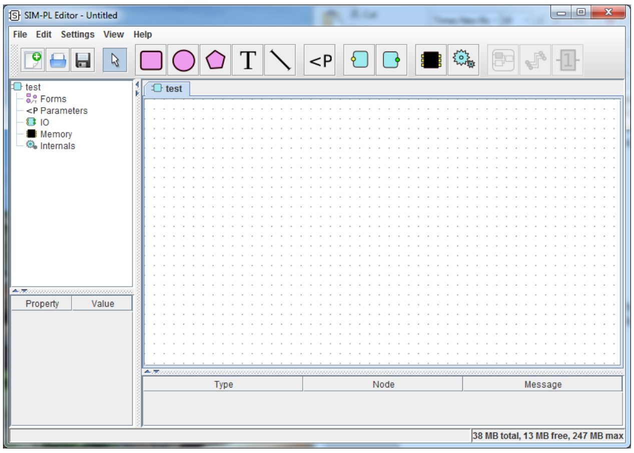


Figure 4. New component dialog window

After the component type and name are provided, click "OK" to create the new component. It results in an empty worksheet and a basic filling of the tree view pane (see figure 5).

2.2.1 Simple and complex components

The difference between simple and complex components is that a simple component is a single component not connected to other components. If a simple component is loaded into the editor only the drawing buttons are enabled. In case of creating or loading a complex component into the editor also the buttons ‘add a subcomponent’ and ‘add wires’ are eneabled. See for details 2.5 and 2.6.

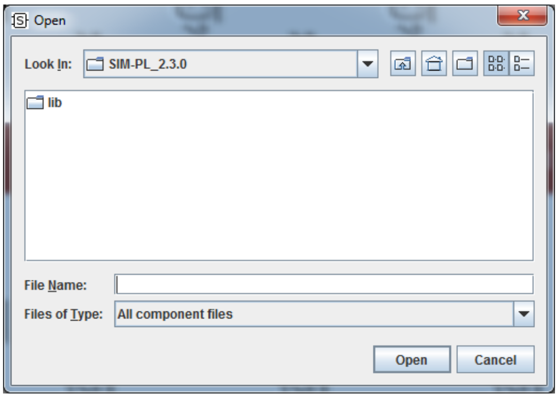


38 MB total, 13 MB free, 247 MB max

Figure 5. New component worksheet

2.3 Opening an existing component

For opening an existing component project click on the button, by pressing Ctrl + o or by using the menu File → Open. One will undoubtedly presented with a file browser dialogue as shown in figure 6.



Look In: SIM-PL_2.3.0

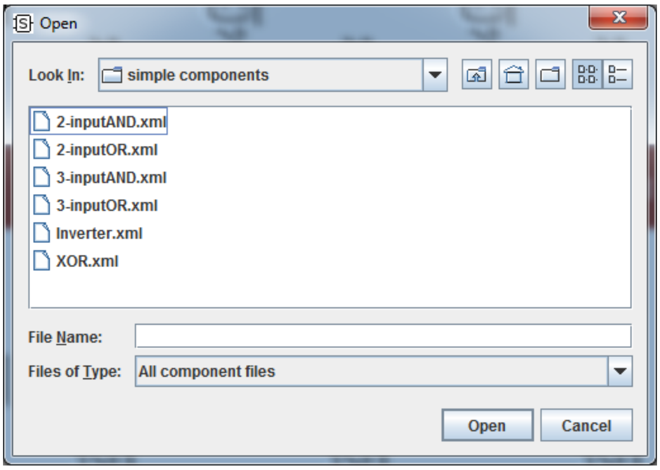
lib

File Name:

Files of Type: All component files

Figure 6. ‘Cold start’ file browser dialogue window

Search for the unzipped ‘SIM-PL Components’ folder and open the ‘simple components’ folder. Now one will undoubtedly find a file browser dialogue comparable with figure 7.



Look In: simple components

2-inputAND.xml
2-inputOR.xml
3-inputAND.xml
3-inputOR.xml
Inverter.xml
XOR.xml

File Name:

Files of Type: All component files

Figure 7. File browser dialogue window listing simple components

Select, for the sake of this example, the project ‘2-inputAND.xml’ and press the "OK" button. The supported file extensions are .xml and .sim-pl. Opening this project presents a worksheet shown in figure 8

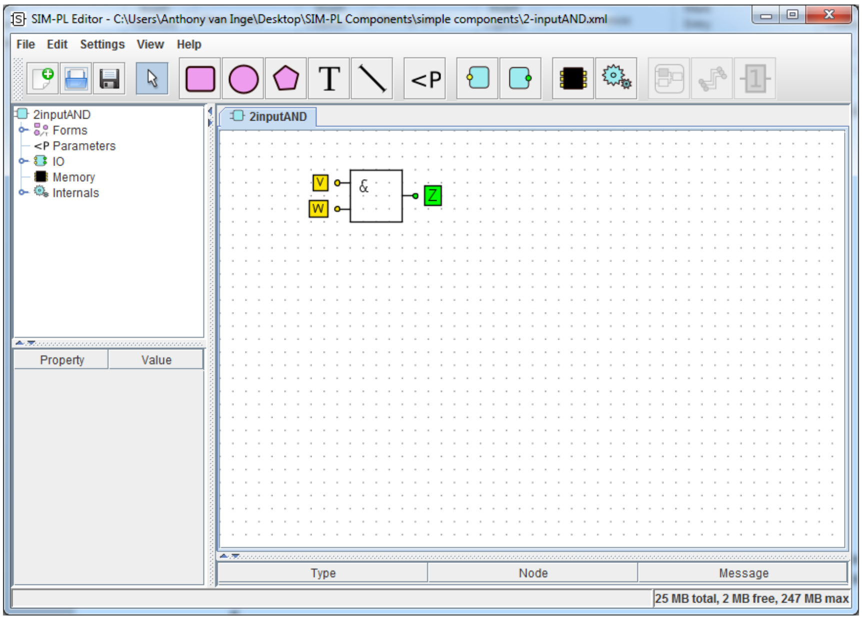

SIM-PL Editor - C:\Users\Anthony van Inge\Desktop\SIM-PL Components\simple components\2-inputAND.xml
File
Edit
Settings
View
Help
<P
2inputAND
Forms
<P Parameters
IO
Memory
Internals
Property
Value
2inputAND
V
W
&
Z
Type
Node
Message
25 MB total, 2 MB free, 247 MB max

Figure 8. Worksheet with a loaded component

2.4 Saving the current component

Save a component by clicking on the 💾 button, by pressing Ctrl + s or by using the menu File → Save. The supported file extensions are .xml and .sim-pl.

2.5 Creating and editing simple components

The following buttons in the toolbar can be used to draw outlines and place text. The buttons represent from left to right: draw rectangles, ellipses, polygons, text, and lines (see Figure 9. Drawing tools buttonsfigure 9).

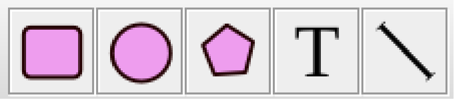


Figure 9. Drawing tools buttons

A drawing created with these tools will not result a functional component, it is still just a drawing. In figure 10 an example of a drawing of an outline of some, yet undefined, component.

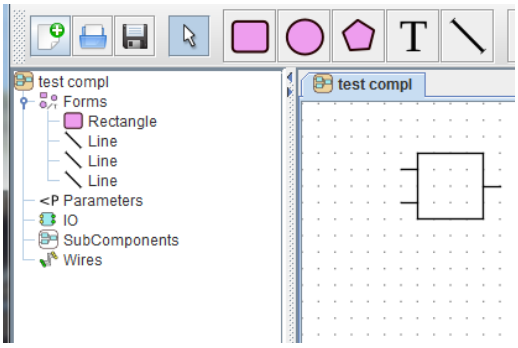


Figure 10. Worksheet with drawing

The first step in promoting a drawing to a component with basic functionality is done by two tools: 'add an input' and 'add an output' (see figure 11).

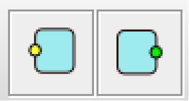
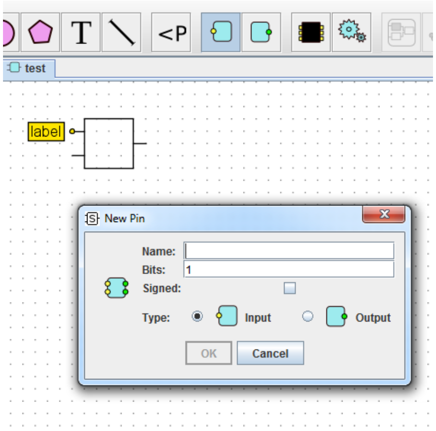


Figure 11. Input & output buttons

Figure 12 illustrates the placement of an input at the stage that a name for the label must be defined.



New Pin

Name:

Bits: 1

Signed: □

Type: ● Input ○ Output

OK Cancel

Figure 12. Worksheet with input placement

Figure 13 illustrates the final placement of inputs and output including labels. In the tree view pane the list of the form and IO elements of this component is visible. Clicking in the list in the tree view pane results in displaying details in the property pane. Also clicking in the worksheet pane selects elements in the tree pane and shows details in the property pane.

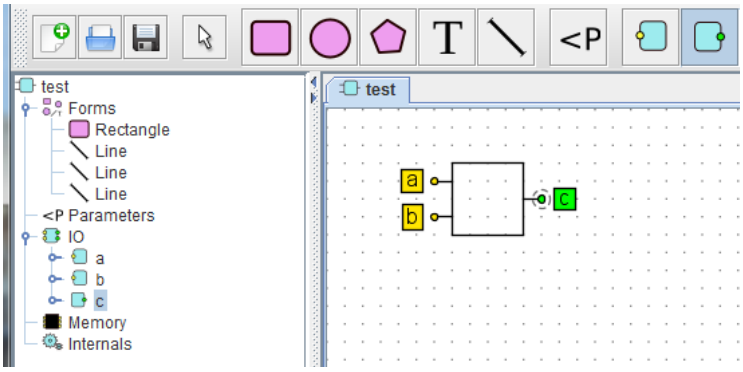
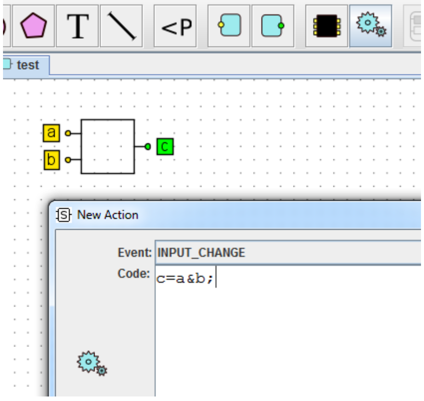


Figure 13. Worksheet with final placement of inputs and output

What is still missing is the function of the component. The button (Program an event) is to add a function to a component. The format of the programming language for describing the functionality is comparable with “C”. In this example an AND function is programmed (see figure 14)



New Action

Event: INPUT_CHANGE

Code:

```
c=a&b;
```

Figure 14. AND function

In the property pane (if ‘INPUT_CHANGE in the tree pane is selected) the program of the component can be edited by clicking ‘Edit Code’. In this window it is possible to parse the program for locating bugs.

2.6 Creating and editing complex components

(see next version of this document)

3 Prologue SIM-PL Simulator.

By double clicking Executer.jar the SIM-PL simulator should open with a window containing a menu and a tool bar (see figure 1). (if not, it is probably related to a Java VM issue)

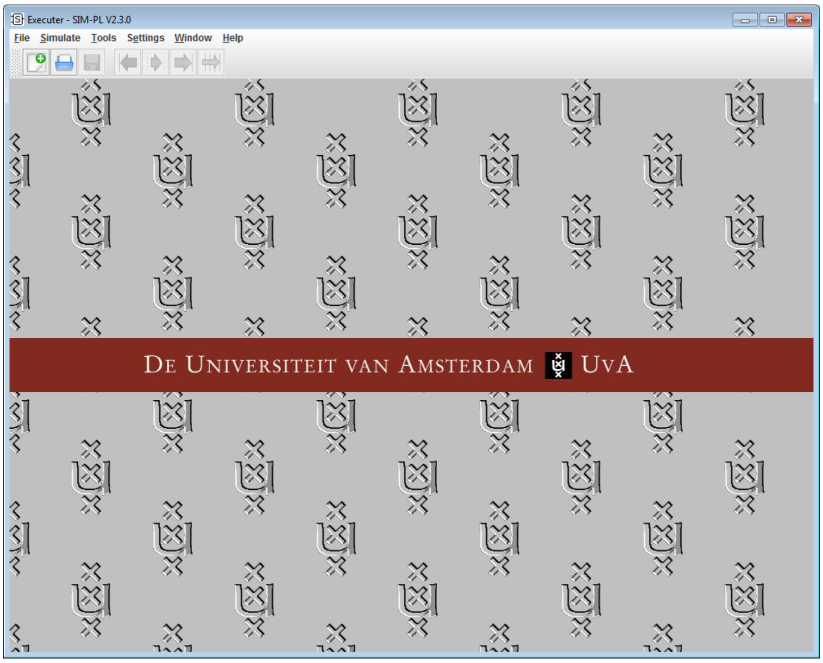
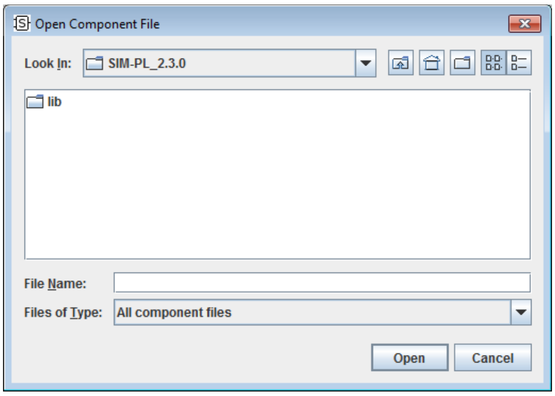


Figure 15. Empty SIM-PL simulator window

For opening a component click on the button, by pressing Ctrl + n or by using the menu File →New. One will undoubtedly presented with a file browser dialogue as shown in.



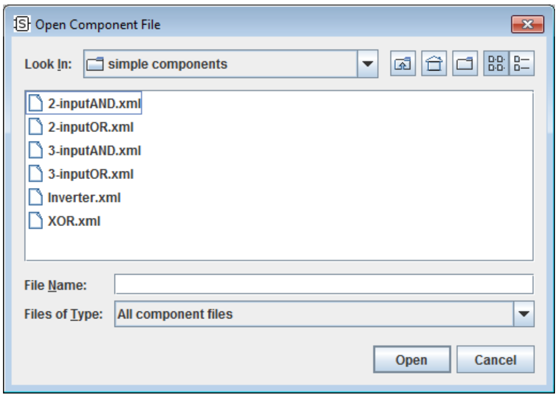
Look In: SIM-PL_2.3.0

File Name:

Files of Type: All component files

Figure 16. ‘Cold start’ file browser dialogue window

Search for the unzipped ‘SIM-PL Components’ folder and open the ‘simple components’ folder. Now one will undoubtedly find a file browser dialogue comparable with figure 17



Open Component File

Look In: simple components

2-inputAND.xml
2-inputOR.xml
3-inputAND.xml
3-inputOR.xml
Inverter.xml
XOR.xml

File Name:

Files of Type: All component files

Figure 17. File browser dialogue window listing simple components

Select, for the sake of this example, the project ‘2-inputAND.xml’ and press the "OK" button. The simulator shows 3 panes:

- Component pane
- Time Sequence Diagram pane (also known as Digital timing diagram)
- Program Editor pane

For the prologue, it is sufficient to treat a very limited number of functions.

The component input and the output values are visible now. By clicking on an input one can change the input. If done one notice that the output will not change and is because there are no simulation cycles made. By clicking several times on the green arrow (Perform a single simulation step) or a single click on the orange arrow (Simulate an entire cycle) the output will change.

In the Time Sequence Diagram pane the signals of inputs and output are plotted. For more insight in the component it is possible to generate a "truth table": choose: Tools → Show Truth Table.

For even more insight one can generate a truth table in the Program Editor pane. Choose File → Generate Truth Table. In the Truth Table window keep the default value (10) and click OK. It will generate code for the component. At the bottom of the Program Editor pane click the Compile button to generate runnable instructions for the component. By clicking on the red arrow (Restart the simulation and load the (compiled) program) the first code line will be highlighted. Now clicking on the orange arrow will execute each code line. Concurrently a digital timing diagram is drawn in the Time Sequence Diagram pane.

4 Appendix A.

The symbols used in this instruction manual follow the standard NPR 5157, see details below

Document	NPR 5157:1989 nl;en
Publication date	1989 02 01
English title	Logic symbols;Symbols for integrated circuits
Nederlandse titel	Logicasymbolen;Symbolen voor geïntegreerde schakelingen
ICS code(s)	01.080.40 (Graphical symbols for electrical drawings, schematics, diagrams, maps and relevant technical product documentation) 31.200 (Integrated circuits. Microelectronics)
Withdrawal date	
Publisher	Nederlands Normalisatie-instituut
Number of pages	276

Nederlandse samenvatting

Deze richtlijn bevat symbolen voor bestaande geïntegreerde schakelingen (IC's) en van enkele welbekende en veel gebruikte IC-families. Deze symbolen zijn bestemd voor gebruik in logica-stroomkringschema's (uitvoeringsschema's).

English summary

This Code of Practice contains symbols for existing integrated circuits (IC's) of some of the most popular IC families. These symbols are meant for use in detailed logic diagrams.

See also the website of the "Nederlands Normalisatie-instituut : http://www.nen.nl/