



SmartSim

Version 1.2.1

User Manual

A Digital Logic Circuit Design and Simulation Package
for Windows & Linux PCs and the Raspberry Pi

Ashley Newson

© 2012

Table of Contents

About This Document.....	4
Copyright.....	4
License.....	4
Website.....	4
Introduction.....	5
System Requirements.....	6
Installation.....	7
Windows OS (Pre-compiled).....	7
Raspberry Pi (Raspbian).....	7
Linux-based OS (Pre-compiled).....	7
Installing From Source.....	8
Getting Started.....	9
Designing.....	10
Designer Window Reference.....	10
General Tips.....	11
Tool Reference.....	11
Example Tutorial.....	13
Hierarchy.....	18
Simulating.....	25
Simulator Window Reference.....	25
Starting a Simulation.....	25
Tool Reference.....	26
Understanding Outputs.....	28
Interacting with a Circuit.....	28
Changing the Simulation Speed.....	28
Stepping.....	28
Exploring Hierarchy.....	28
Logic Timing Diagrams.....	29
Projects.....	31
Saving.....	31
Loading.....	31
Sharing Projects.....	31
Exporting and Printing.....	32
Exporting.....	32
Printing.....	32
Built-in Component Reference.....	33
Buffer.....	33
And.....	33
Or.....	33
Xor.....	33
Multiplexer.....	34
Tristate Buffer.....	34
Toggle.....	34
Reader.....	34
Constant.....	34
Positive-Edge-Triggered D Flip-flop.....	35
Positive-Edge-Triggered T Flip-flop.....	35
Memory.....	35
Basic Seven Segment Display.....	36

Errors, Warnings, and Solutions.....	37
Program Start-up Errors.....	37
Design Errors.....	37
Validation Errors.....	37
Circuit Warnings.....	38
Memory Chip Errors.....	38
Simulation Errors.....	39
Saving and Loading Errors.....	39

About This Document

Copyright

SmartSim User Manual, Copyright Ashley Newson 2012. See license below.

License



SmartSim User Manual by [Ashley Newson](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License](#).

(<http://smartsim.org.uk>)

(http://creativecommons.org/licenses/by-nc-sa/3.0/deed.en_GB)

Website

SmartSim website: <http://smartsim.org.uk>

Introduction

SmartSim is a cross-platform digital logic circuit designer and simulator. It is designed to be used as a learning aid to digital logic. SmartSim has a powerful hierarchy feature which is designed to enable the creation of complex circuits by allowing users to create their own components and reuse them many times over.

This user manual corresponds to SmartSim version 1.2.1, and contains instructions on how to use this software and get creating logic circuits. It also contains a reference of the components available for use within the software. This user manual itself, however, is not meant as a learning resource for understanding digital logic.

Not all screenshots in this manual may be up-to-date, as SmartSim is still under development, but they should still be fit for purpose as a demonstration or an explanation.

System Requirements

Item	Requirement
Operating Systems and Processor Architectures*	Any of: <ul style="list-style-type: none">• Windows XP or later<ul style="list-style-type: none">◦ x86• Linux based<ul style="list-style-type: none">◦ x86◦ x86_64◦ ARM
Free Disk Space	25 MB (including libraries)
Free Memory**	25 MB
Peripherals	<ul style="list-style-type: none">• Mouse, touch screen or other pointing device• Keyboard• Monitor or other display• Optional printer

*Only the operating systems and processor architectures on which the software has been tested have been listed.

**Free memory refers to additional memory on top of what is required by your specific operating system and other running programs. Ultimately, the amount of free memory required is dependant on how complex your design is, and only a recommended amount is given.

Installation

Windows OS (Pre-compiled)

The installation file “smartsim.zip” is a compressed zip file containing the software, its resources, and DLL dependencies.

In order to extract the contents of this zip file, you will need a program capable of extracting zip archives. Windows XP and later come packaged with support for zip folders by default, so it is not necessary to install this software yourself. The process of installing a zip extractor is not detailed in this manual.

Assuming you can use Windows' zip archive manager:

1. Open up “smartsim.zip” just like you would as if it was a normal folder by double clicking on it.
2. Select the sub-folder “smartsim” and copy it (using Ctrl-C).
3. Navigate to the location you wish to install SmartSim to and enter it.
4. Paste the copied folder (using Ctrl-V).

You may uninstall SmartSim at any time by deleting the “smartsim” folder.

Raspberry Pi (Raspbian)

All you should need to do to install SmartSim is run the commands “apt-get update” and then “apt-get install smartsim” as root in a terminal (try “sudo apt-get update” and “sudo apt-get install smartsim”).

Linux-based OS (Pre-compiled)

The installation file “smartsim.tar.gz” is an Gzip compressed tar archive containing the software and its resources. You should download one appropriate for your system's architecture.

In order to extract the contents of this tar file, you will need a program capable of extracting tar archives with support for gzip compression. Tar archivers are usually packaged with Linux distributions, but if you do not have one you must install one using your package manager or by other appropriate means. The process of installing a tar archiver is not detailed in this manual.

Before being able to run the software you will need to install the software dependencies for the program by using your package manager or by other appropriate means. The following packages and their dependencies need to be installed on your system:

- GTK+ 2
- LibRsvg

You may extract the tar file in any way you wish, but for universal guidance only the command-line based method is detailed.

1. Place the “smartsim.tar.gz” archive in the directory you wish to install to.
-

2. Open up a terminal and navigate to the installation directory. (You can use the command “cd” followed by the directory name to change directory. Use the command “ls” to display the contents of a directory.)
3. Use the command “tar -xf smartsim.tar.gz”.

To uninstall SmartSim, simply delete the extracted “smartsim” folder.

Installing From Source

This section assumes you are using a Unix or Unix-like system capable of compiling and installing projects using Autotools. This method is primarily intended for advanced users.

The source code for SmartSim can be acquired using Git with the command:

```
git clone https://github.com/ashleynewson/SmartSim
```

Alternatively, if you do not have git installed, you can download the repository as a zip archive at:

```
https://github.com/ashleynewson/SmartSim
```

In a terminal, enter the new SmartSim source directory containing a script called “configure”.

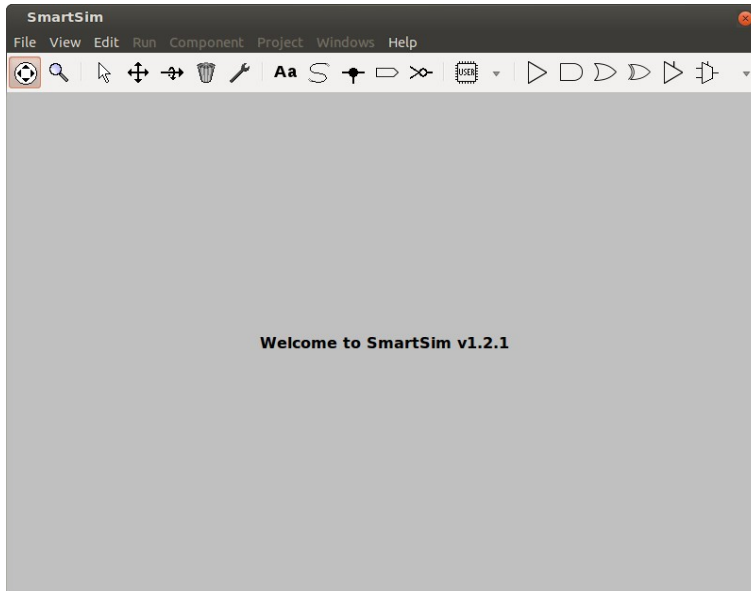
Type the command “./configure ; make” to build SmartSim, and “make install-strip” to install SmartSim on your system.

If you wish to install SmartSim into a location other than “/usr/local”, specify this to the “configure” script. For example, to install instead to “/usr”, use “./configure --prefix=/usr”. If you wish to instead create an installation which runs in the current working directory, use the prefix “.”.

Getting Started

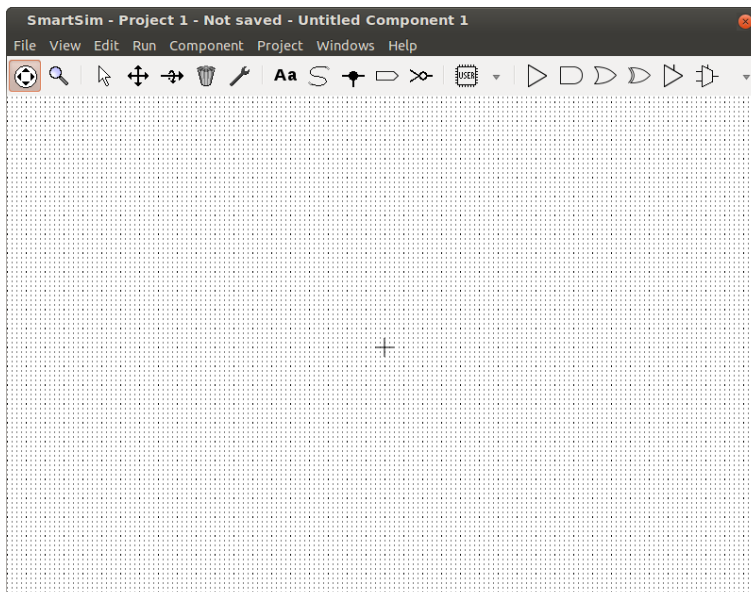
To start SmartSim, simply execute “smartsim.exe” (Windows) or “smartsim” (Linux), located within the installed “smartsim” folder.

Upon start up, you will be greeted with a window as shown:



To begin you must either create a new project via the menu bar using *File – New Project*, or instead open an existing project using *File – Open Project*.

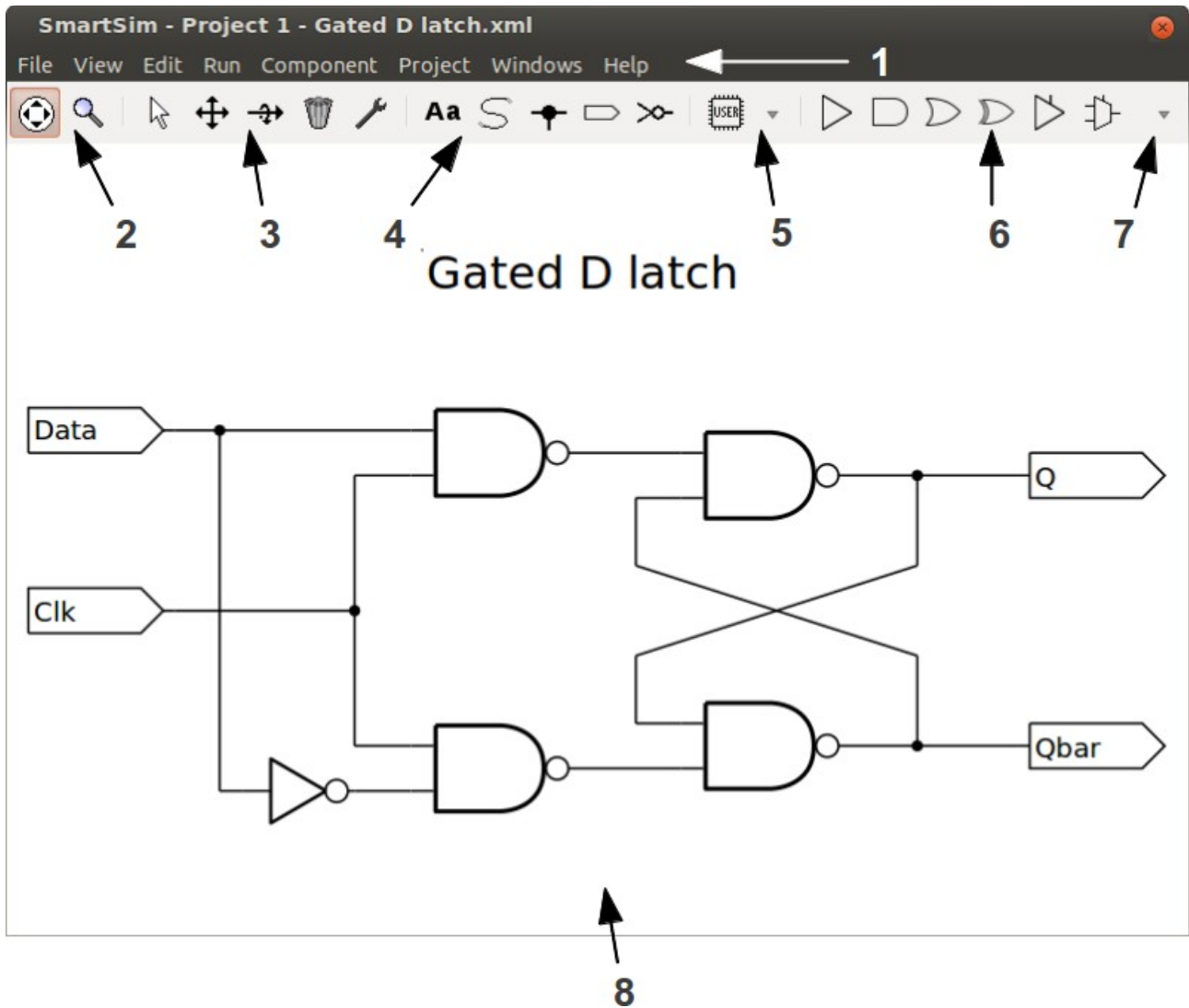
When creating a new project, or expanding an existing one, you may either create a new component using *File – New Component* or open existing components with *File – Open Component*.



You are now ready to begin designing.

Designing

Designer Window Reference



1. Menu bar
2. View controls
3. Manipulation tools
4. Miscellaneous tools
5. Insert custom component
6. Insert built-in component
7. Extra tools and components drop-down menu
8. Work area

General Tips

To aid consistent layout in your circuit design, you can show/hide the grid by selecting *View – Show Grid* from the menu bar.

To make your circuit easier to view with different window sizes, you should create your circuit in the default centre of the work area, which is shown with a cross-hair when the grid is on. This will reduce the need for scrolling, especially when the circuit is viewed on differently sized screens.

Tool Reference

To use any of the following tools, simply click on the icon shown and perform the described actions. Tooltips are provided for each tool.

Scroll



The scroll tool is used to move the user's view of the design. Click and drag up, down, left, or right in the work area to scroll the viewable area.

Zoom



The zoom tool adjusts the display scale used to view the design. Click and drag vertically to adjust the zoom (drag up to zoom out, drag down to zoom in).

Select



The select tool is used to select components, wires, and annotations.
(It has no current function, and is meant for future development.)

Move



The move tool will move a component, wire, or annotation. Click and drag an object to move it. Moving a component disconnects it from any wires.

Orientate



The orientate tool can change the direction of a component or flip a component. To change the direction of a component click on a component and drag up, down, left, or right. To flip a component, simple click on it without dragging.

Changing a component's orientation (direction or flip) disconnects it from any wires.

Delete



The delete tool will delete a component, wire, annotation, or tag. Click on an object to delete it.

Adjust



The adjust tool can alter various properties of objects. Click on an object to change its properties.

When used on an annotation, you can change its text and font size.

When used on a component, the available options vary. You can alter the number of pins on some components, such as primitive logic gates. Special components, such as clocks, will take their own specific properties. Adjusting the number of pins on a component will disconnect it from any wires.

When used on a wire, you can set the initial signal in the simulation.

When used on a tag, you can change the label, flow, and ID of that tag.

Annotate

Aa

The annotate tool will add an annotation with specified text and font size. Click where you wish the top left-hand corner of the annotation to be placed.

Wire



The wire tool is used to lay wires in a circuit. To begin adding a wire, click on the desired starting point. Continue drawing the wire by clicking on each point of the wire. When you have finished drawing, click on the last point drawn. You may undo a point by clicking on the previous point.

Most circuit diagrams are conventionally drawn using horizontal and vertical lines, but diagonal lines may be drawn if required.

Wires which are being drawn are displayed grey, whereas drawn wires are black.

To cancel the wire currently being drawn, without undoing each individual point, either using the delete tool on the being-drawn wire, or click on the wire tool again with the tool already selected.

Bind



The bind tool is both used to connect different wires together and to connect wires to component pins. To bind, click on the point where the items meet. You can unbind by simply clicking on an existing binding.

Note: Most binding operations are done automatically (Automatic Binding), but sometimes, you might need to bind or unbind objects manually.

Tag



The tag tool creates and removes interface tags on wires. Tags are used to define a circuit's interface signal names and types (input, output or bidirectional). Click and drag from or to a wire to create or edit a tag (details about using tags is given in the hierarchy section). Click without dragging on a wire to remove its tag.

Invert



The invert tool will invert the signal going into or out of a component's pin. To toggle invert, click on the end of a component pin. Any pin, input or output, can be inverted, on both built-in and custom components alike. For example, this is how an AND gate is converted into a NAND gate, by inverting the AND's output pin.

Insert Custom Component



The insert custom component tool allows you to insert any custom component within the same project. Select a custom component from the drop-down list and click where you wish to insert it. Dragging whilst placing the component will determine its direction. Clicking on the button (not the drop-down list) will select the last used custom component.

Insert Built-in Component

The insert built-in component tools allow you to insert any built-in component. After selecting a component click where you wish to insert it. Dragging whilst placing the component will determine its direction.

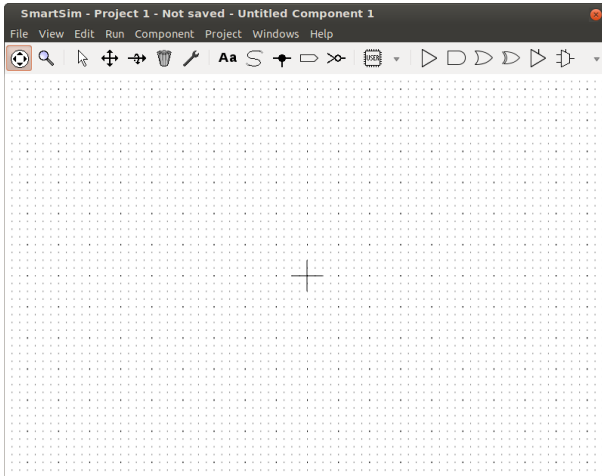
Example Tutorial

The following tutorial will describe the process of creating a simple XOR circuit. You may wish to look at the built-in component reference for information on components used. This tutorial will cover:


- Inserting components
- Drawing wires
- Inverting pins

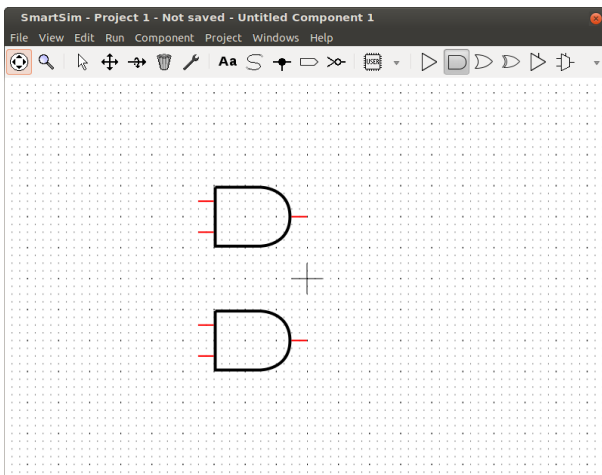
To begin creating a component, start up the application and choose *File – New Project*, then *File – New Component*, as described in the Getting Started section.




You may wish to zoom in by selecting the zoom tool and then clicking and dragging down in the work area.




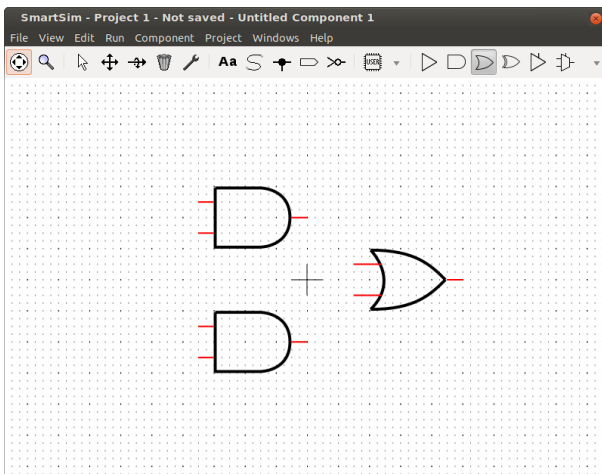
Inserting Components

Select the insert AND gate tool () and click in the work area to insert an AND gate. The direction you approach the point with affects the rotation of the component. You can disable this feature by disabling *Edit – Shadow Components*. You will need to place two AND gates down as follows:



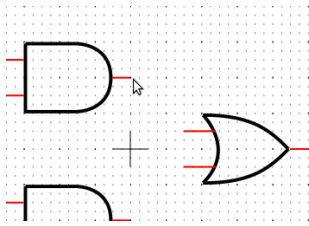
If you make a mistake, you can move components around with the move tool () , change their direction with the orientate tool () , or delete them with the delete tool () . See the Tool Reference section for details on how to use these.

Next, you need to place down an OR gate () . Place it as follows:

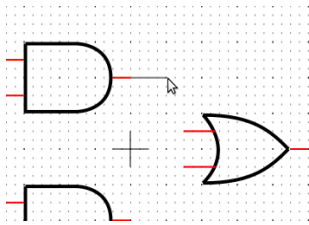


Drawing Wires

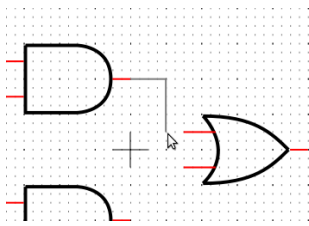
You have placed down all the components required to form the XOR, but they still need linking up. So, next you must lay down the wires. Select the wire tool (S). Drawing a wire uses a connect-the-dots method. Start by clicking on the first point of the wire, then click on each subsequent point, finishing a wire by clicking again on the last point specified. If you make a mistake, click on the previous point to undo the last point, or click again on the wire tool to start over. If you finalised a bad wire, use the delete tool (trash icon) on it. When the wire drawing has finished, it will turn from grey to black. Click in the sequence shown:



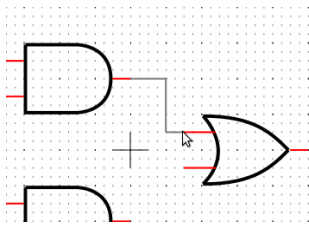
Click on a point to begin the wire.



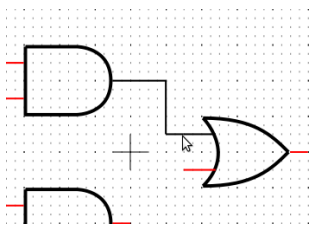
Click on the next point of the wire.



Next point.



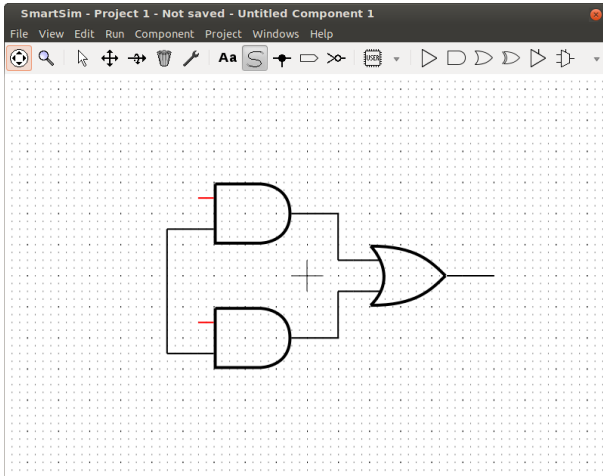
Click on the final point.



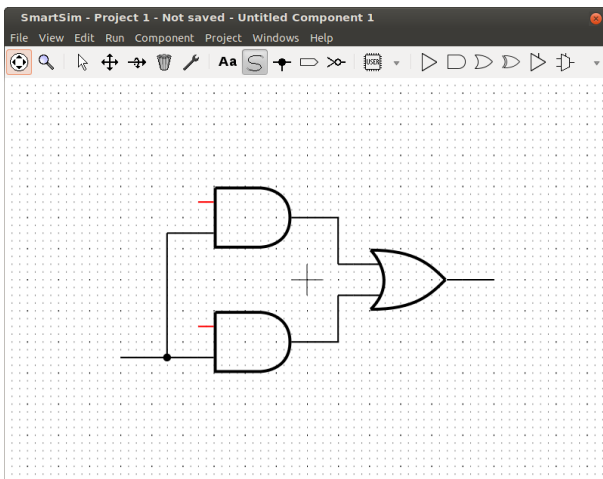
Click again on the final point to finalise the wire.

Wires automatically join to pins when they are created. An unconnected pin is shown in red, whereas a connected pin is shown in black.

Create some more wires to get the following design:

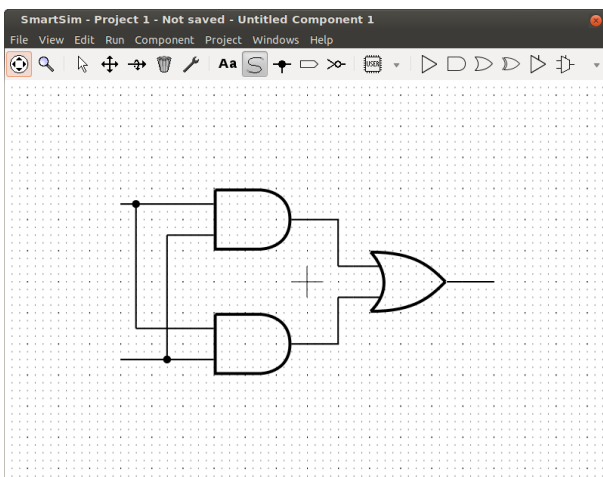


Now place a wire with one of its ends touching the bottom-left wire as shown:




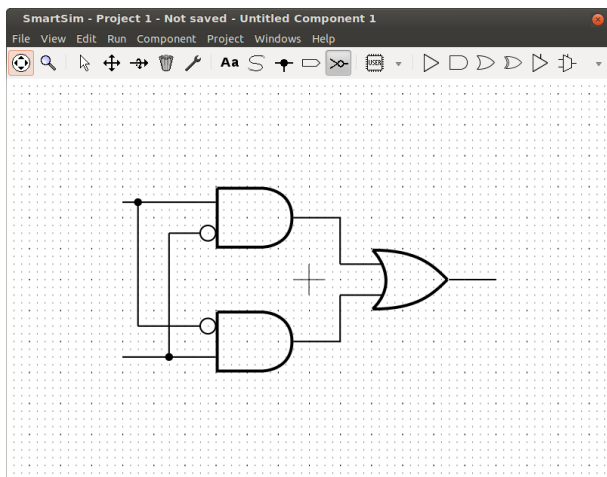
These wires will join, and their connection is shown by placing a small black circle at the junction.

Repeat this as follows:



Inverting Pins

The wires are now connected to the components, but some of the pins must be inverted. To invert a pin, select the invert tool () and click on the end of the pin (the end which is away from the component). Do this to end up with:




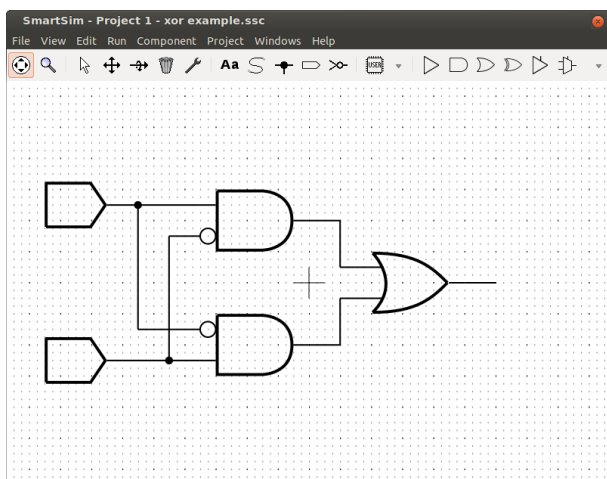
The actual XOR circuit is now complete. The only thing which is needed for this to work in a simulation is some form of input and output.

You may wish to save your current component so that you can use it in the hierarchy tutorial. Go to *File – Save As* to bring up the save dialogue. Save the component with a memorable file name, such as “xor example.ssc” in a folder of your choice.

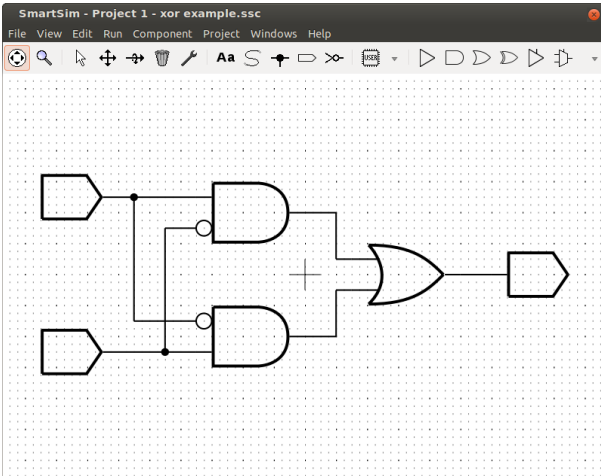
Inputs and Outputs

For the remainder of this tutorial, we will not use hierarchy, and instead just add on some Toggle Buttons and an output Reader.

Insert two Toggle Buttons () so that their pins are just touching the wires on the left. If you cannot see the insert Toggle Button tool, your window may be too small to display it – either resize the window or click on the drop-down box to right of the tool bar.



After you have inserted the Toggles, insert a Reader () touching the wire on the right:



This will produce the complete circuit.

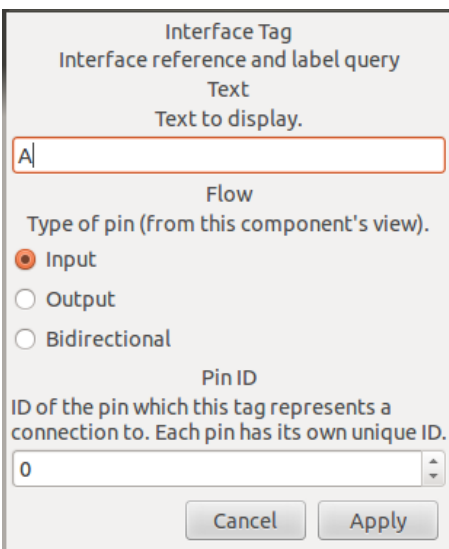
If you wish to simulate this circuit, read the section on Simulating.

Hierarchy

Hierarchy is a powerful tool which allows a user to create complex circuits by creating their own sub-circuit components, which can then be used within other components. One could, for example, create an adder component and reuse it many times.

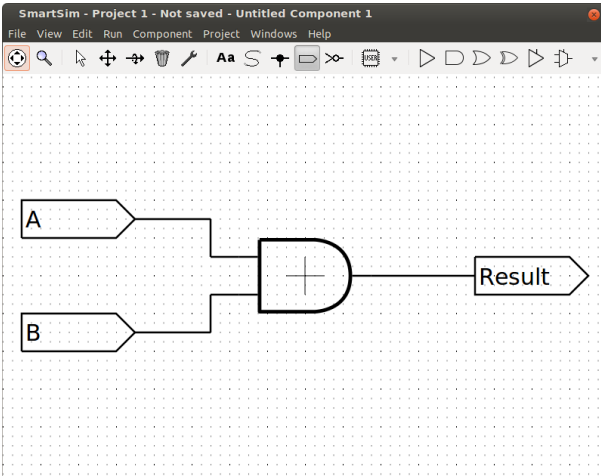
Tagging Wires

An interface tag is used to specify where a signal enters or leaves a custom component. Each tag must have its own ID which is unique to that component. Tagging is done with the tag tool (☐). To use the tag tool, first select it, then click and drag to or from a wire. This will present you with a properties dialogue:



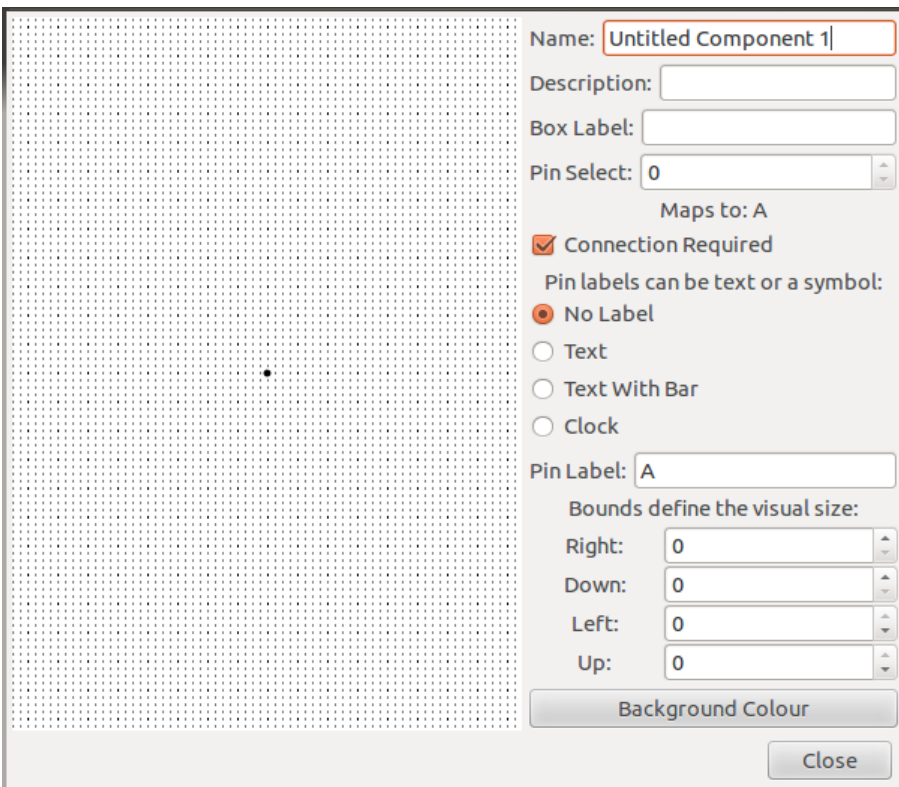
You can enter in a label which is displayed inside the tag, whether it is an input, output or bidirectional tag, and also the pin ID. When creating a new tag, the lowest free ID is automatically assigned.

The appearance of tags is shown below, where “A” and “B” are inputs and “Result” is an output:



Customisation

In order to use a custom component in a design, the custom component must be customised. Customisation is the process of describing how the component should appear when used in other components. Customising requires that all interface tags are defined with unique IDs. Customise a component using *Component – Customise* from the menu bar. This will show the customiser dialogue:



The name of a component is used as an identifier for when it is used in other components.

The description is only used for the benefit of the user. It does not affect the designing process.

The box label is the text which is displayed inside the box representation of the component.

The display shows the box representation of the component. Additionally it is used to set the position of pins. Click where the end of the selected pin should be to define a pin's position.

Pins can be selected using the spin button. By default all pins require a wire connection, however this can be overridden by unchecking “Connection Required”. Various label types are available for individual pins. The selected pin is displayed in the display area as blue rather than black.

Bounds specify the size of the box representation.


The Background colour button can be used to change the box fill colour.

You should first specify the name, description, and box label of the component. Afterwards, specify the bounds. Define the pins last.

Once you have customised your component, it may be inserted into another component.

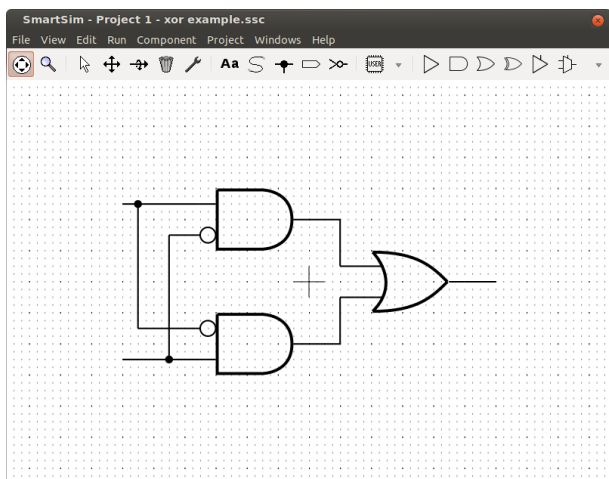
When assigning IDs to interface tags and pins, you should ideally have consecutively numbered pins next to each other in the box design. Make sure you get the box design right the first time, as changing the design after it has been integrated into another component can be very difficult and time consuming.


Inserting and Using a Custom Component

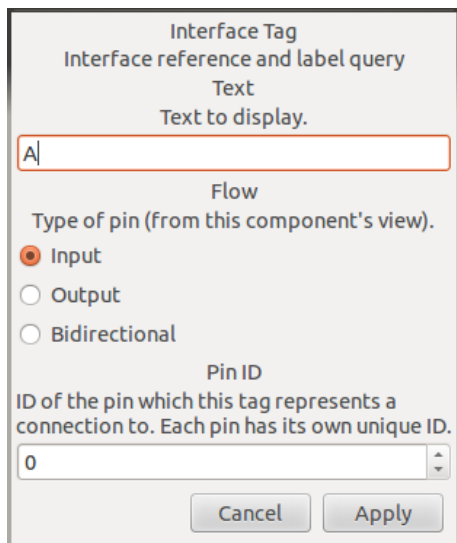
Inserting a custom component is similar to inserting any other component. You must select the component you wish to insert from the drop-down by the insert custom component tool (). Insert it by clicking in the design area. Clicking on the insert custom component button (not the drop-down) will select the last custom component used.

Hierarchy Tutorial

This tutorial will use the XOR created in the design tutorial. It is assumed that the XOR has been developed as follows (see earlier tutorial):

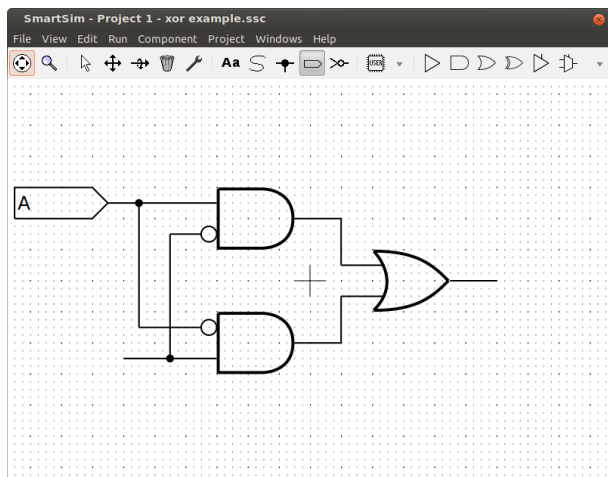


The first step is to add interface tags onto the wires using the tag tool (). Select the tag tool and click and drag from two dots left of the upper left wire to end of the wire. You will be presented with a dialogue:

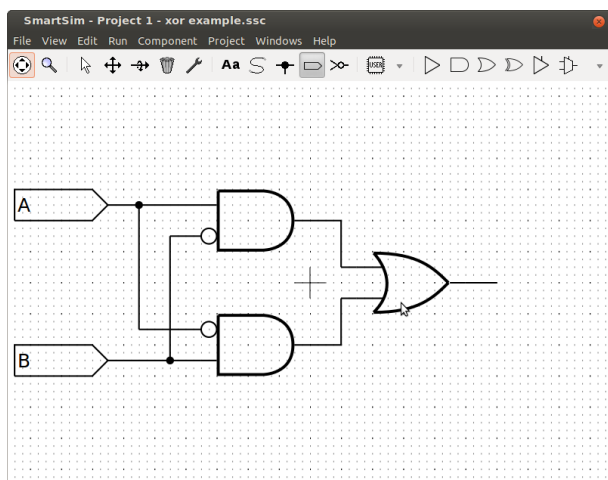


Change the text to A, and ensure the other entries match the above dialogue and click Apply.

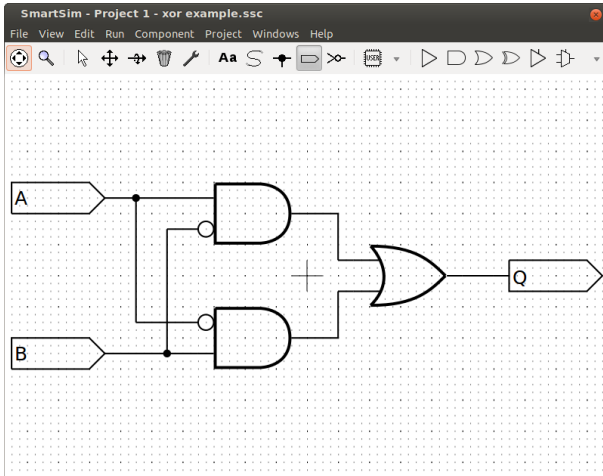
You should be left with:



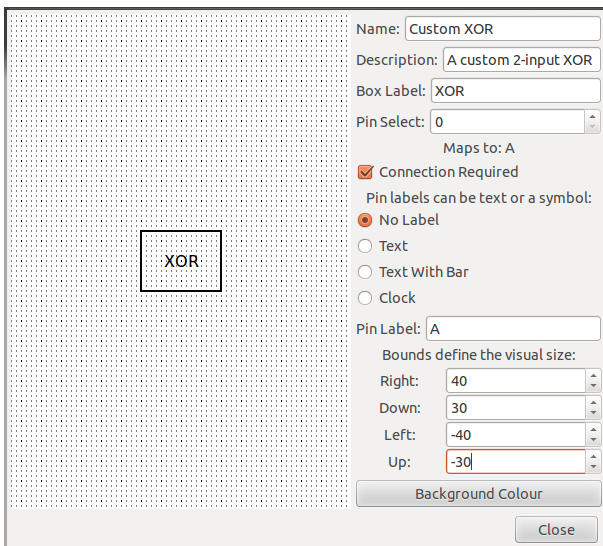
Repeat this for the lower-left wire, but instead label it "B" rather than "A". Leave the ID as 1.



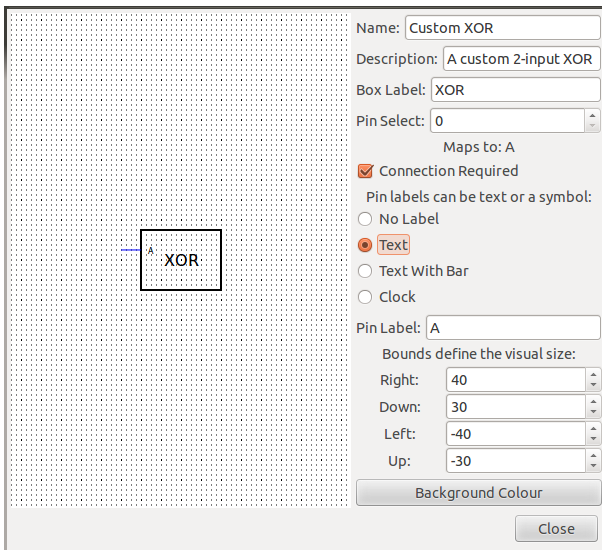
With the final right-most wire, drag outwards two dots right from the end of the wire. This will automatically make it an output tag. Label this tag as "Q". Leave the ID as 2.



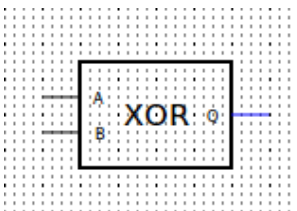
The tag definitions are complete. Now the component must be customised. Open the customiser dialogue through the menu bar using *Component – Customise*. Fill in the customiser dialogue as follows:



Next, define the pin layout. The currently selected pin is pin 0. Click on the radio button which says “Text”. The pin label is by default set to the text of the tag it relates to (you do not need to change this). To specify the position and length of a pin, click on where the end of the pin should be in the display area. Set pin 0 (“A”) as shown:

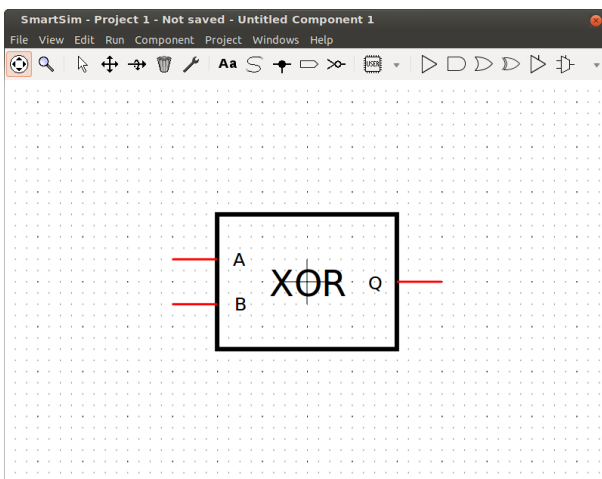


Do this for the other pins, by cycling through the pin selector, to get the following box design:

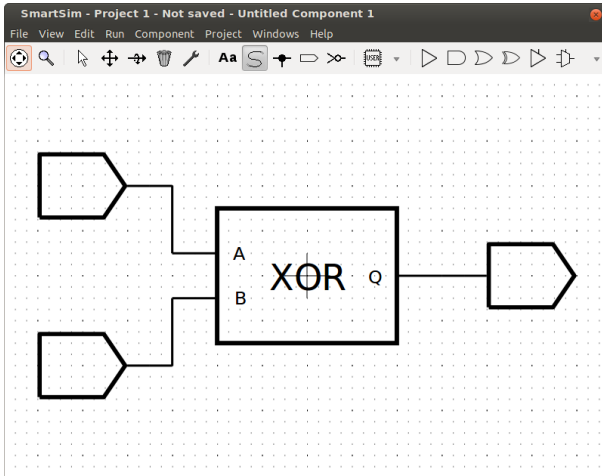


Close the customiser dialogue. You now have a component which is ready to be inserted into another component.

You need to create the high level component, which will use the custom XOR component you have created. Create a new component with *File – New Component*. In the new component, use the insert custom component tool (USER) drop-down and click “Custom XOR”. Click in the work area to create an instance of your custom XOR.



Finally, create some Toggle Buttons and Reader and connect them up with wires to the XOR, just as you did in the previous tutorial.



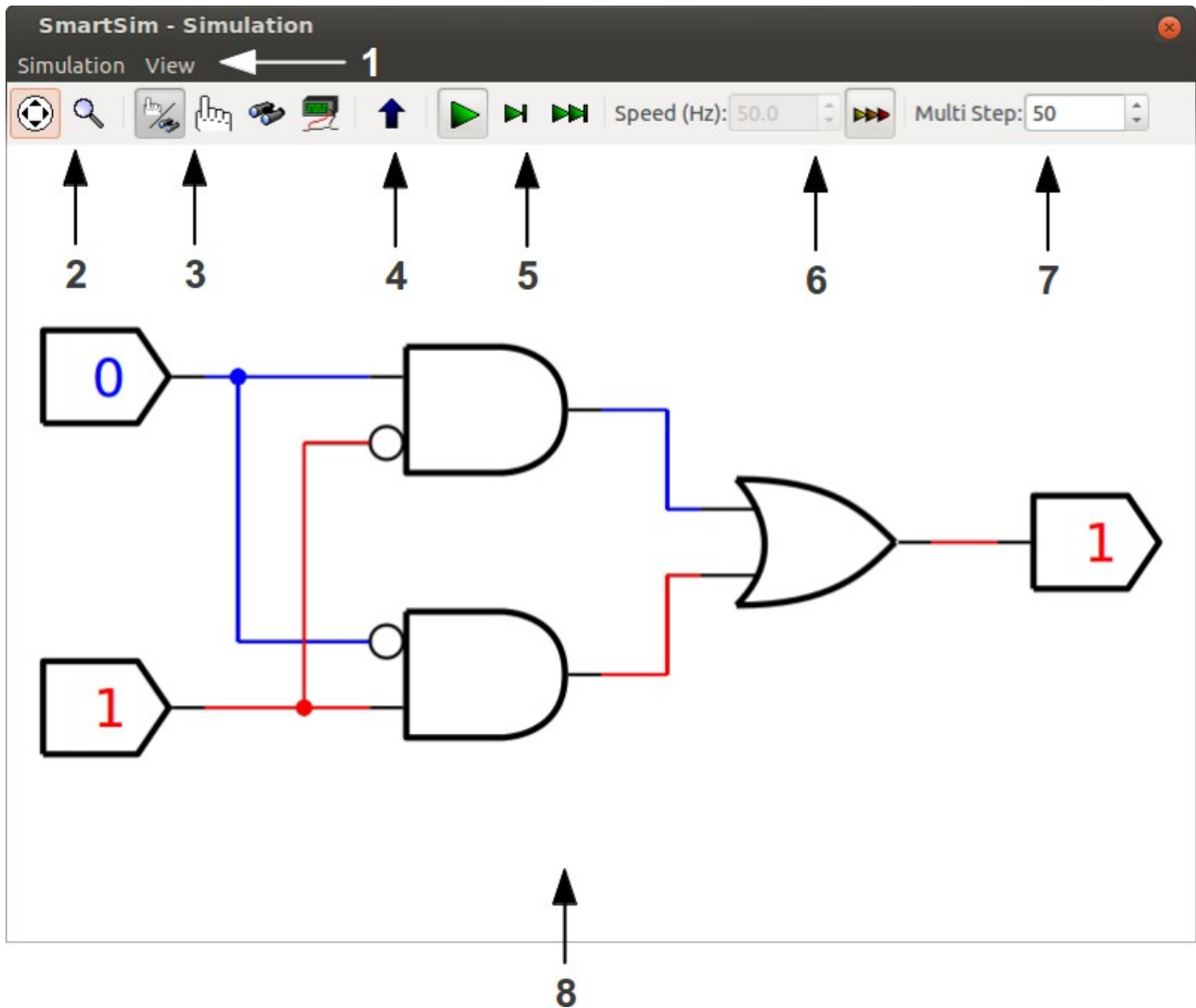
If you wish to simulate this circuit, take a look at the section on [Simulating](#). Before you can simulate, the circuit must first be set as the root component using *Component – Set as Root*. This XOR should behave just as it did in the earlier tutorial, but you can now reuse it more easily.

Loading a Hierarchical Design Without a Project

In order to load a saved hierarchical design without the aid of a project, you must load the lowest-level components first (the contained rather than the containers). If you try to load a component which requires a component which is not already open, it will raise an error stating which component is required. Usually this is not a necessary process, as you can use projects to manage your components.

Simulating

Simulator Window Reference



1. Menu bar
2. View controls
3. Mouse tools
4. Shrink tool
5. Simulation run/pause controls
6. Simulation speed controls
7. Multi step size control.
8. Simulation area

Starting a Simulation

In order to simulate a design, you must specify something called the Root Component. The root

component is effectively the component within your project which you are simulating, i.e. the highest level component. You would prefer to simulate a whole processor rather than just one of its sub components. To specify the root component, bring up the window of the component you wish to make the root component, and select, via the menu bar, *Component – Set as Root*.

To start the simulation, choose *Run – Run* from the menu bar. If you would prefer the simulation paused on start up, select the menu item *Run – Start Paused* first. This will bring up the simulation window.

If there are errors preventing a circuit from running, check the “Errors, Warnings, and Solutions” section.

Tool Reference

Scroll



The scroll tool is used to move the user's view of the running circuit. Click and drag up, down, left, or right in the work area to scroll the viewable area.

Zoom



The zoom tool adjusts the display scale used to view the running circuit. Click and drag vertically to adjust the zoom (drag up to zoom out, drag down to zoom in).

Context



The context tool behaves like the interact tool when used on an interactable component, or like the explore tool when used on a custom component. See interact and explore tools.

Interact



The interact tool can be used to change the state of an interactable component, such as a toggle button, in order to interface with the circuit at runtime. Clicking on an interactable component changes its state.

Explore



The explore tool can be used to view the inside of a custom component at run time, so that the sub-components can be seen. Click on a custom component to view its circuitry. You can further click on any custom sub-components.

Click on the background to return to the higher level (container) component.

You can also use the shrink tool to return to the higher level. See the shrink tool.

Watch



The watch tool is used to mark wires to record on a logic timing diagram. Click on a wire to add it to the timing diagram. You will be asked the name of the wire, which will be displayed in the timing diagram.

Shrink



The shrink tool can be used to return to the higher level (container) component when exploring using the explore tool. Clicking on the button returns to the immediately higher level.

Run



The run tool is used to start and pause the simulation. Clicking on the button will toggle running or paused.

Single Step



The single step tool is used to run for a single simulation step and then immediately pause. Clicking on the button will step once and pause, even if previously running.

Multi Step



The multi step tool is used to run the simulation for a specified number of steps and then immediately pause. Clicking on the button will perform the steps and then pause, even if previously running.

The number of steps to perform is by default 50, although this can be changed in the timing dialog. See the timing tool.

Maximum Speed



To toggle between running at the specified simulation speed and running as fast as your computer can handle, click this button.



Understanding Outputs

When the simulation is running, you will be presented with a live view of the signals passing through wires. Different wire colours mean different things.


Colour	Description
Blue	Wire which is set to binary 0 (false).
Red	Wire which is set to binary 1 (true).
Green	Wire which does not have a live input (high impedance, “Z”, state). (This can also be as a result of a design fault whereby a wire does not have any input connections.)
Yellow	Wire which has a multiple input error.

Certain components, such as Constants, Toggles, and Readers, may display “0”, “1”, or “Z” to represent their state.

Interacting with a Circuit


To enter interact mode, select the Interact tool () or the Context tool (). Click on components to interact with them. Not all components can be interacted with.

Changing the Simulation Speed



You can toggle whether the current simulation is running either via the menu *Simulation – Run* or by using the Run toggle button (.

To change the speed at which the simulation is running, use the speed spin-button (labelled “Speed (Hz)”). This will allow you to set the number of simulation steps per second.





A higher value means a faster simulation. Click on the Maximum Speed button () to disable the delay and run as fast as your computer can handle. Slowing the simulation of a circuit's operation can be useful for demonstrating or debugging a circuit design.

Stepping

If you wish to progress through the simulation a step at a time, you can step through using the Single Step () and Multi Step () tools. Single-stepping allows you to progress the simulation by a single simulation iteration, whilst multi-stepping allows you to progress by a number of simulation iterations at a time, where the amount is specified by the spin button labelled “Multi Step”.

Exploring Hierarchy


You may have custom components in your root component, and they themselves may also contain other custom components. To view what is going on within these custom components, enter explore mode by select the Explore tool () or the Context tool (). Click on a custom component to explore inside it. Click on the simulation area background to return to viewing the parent

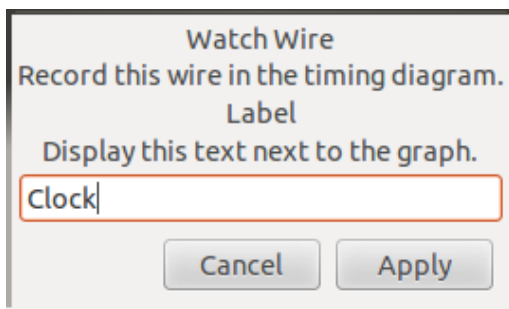
component.

You may explore down any number of levels of hierarchy to reach a specific component instance, and back up again.

Logic Timing Diagrams

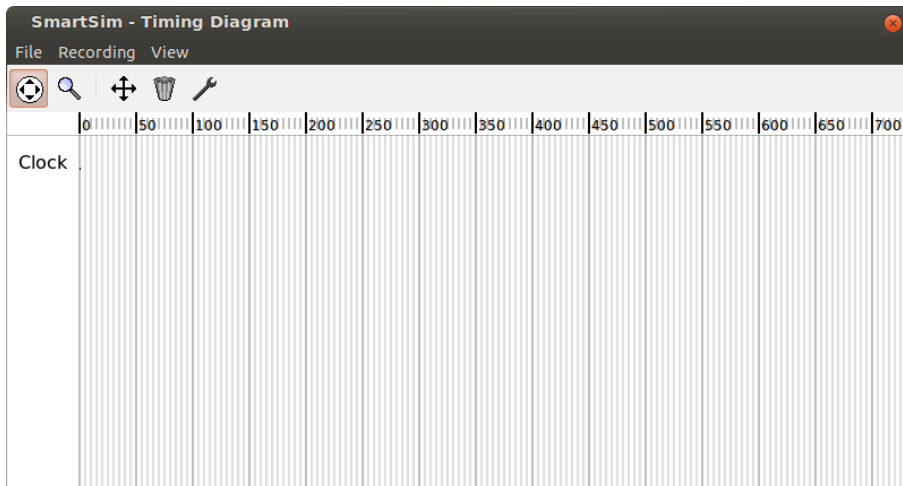
Note: when setting up a circuit for recording in a timing diagram, it is recommended to start the simulation paused, using *Run – Start Paused* in the designer window. You may also wish to turn down the simulation speed, or use stepping when running the simulation.

You can create logic timing diagrams from a running simulation. In the simulation window, select the Watch tool () and click on a wire you wish to record on the timing diagram. You will be presented with a dialogue, asking for the wire's signal label.

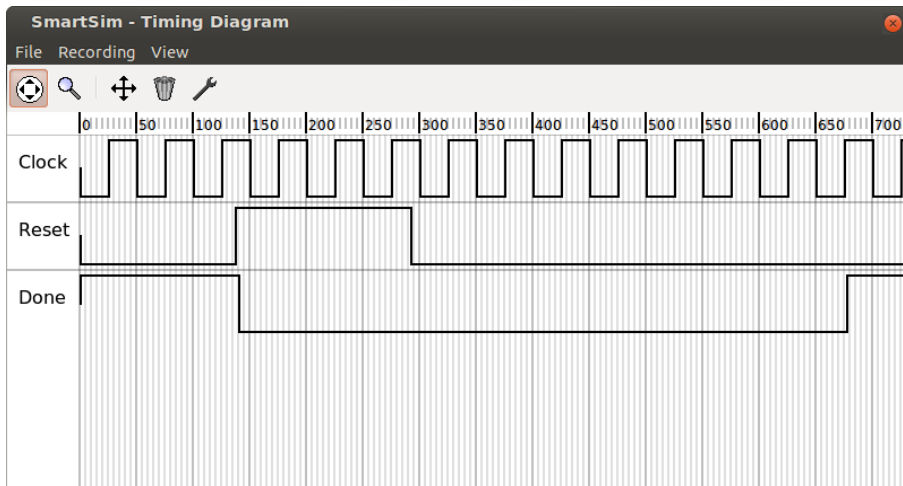


The label you enter will be show at the side of the wire's history graph.

To view the diagram, use the simulation window's menu option *View – Show Timing Diagram*. The timing diagram window is by default set to stay above all windows, but you can disable this with the timing diagram window's *View – Always On Top*.



When you are ready to record, after adding any more wires, you can start the simulation running using the simulation window's Run tool. This will produce graphs for each of the wires being recorded.



If your timing diagram appears blank, this may be because your simulation was already running, and the graphs appear later than step 0. To clear the recording history, and start recording at time 0, select *Recording – Reset*.

You can scroll and zoom the graphs using the Scroll (🔄) and Zoom (🔍) tools. You can change the height order of the graphs by clicking and dragging with the Move tool (↕). You can remove a trace from a diagram, losing its entire history, by using the Delete tool (🗑️). You can change the text of a label by using the Adjust tool (🔧).

Projects

Projects allow you to manage your components more easily. Projects allow you to load all of your components in one go, rather than opening them all individually. A project is self contained, and does not interfere with any other projects. A new project can be created using *File – New Project*.

Saving

Saving a project requires that all the components of your circuit are saved, and this further requires that there are no cyclic dependencies in your circuit design. All these checks are performed when trying to save a project.

You can save a project using *File – Save Project*.

Loading

Loading a project requires that all component files are present and have not been moved.

You can load a project using *File – Open Project*.

Sharing Projects

If you wish to distribute a project, you must also distribute all the components it uses. It is easiest to do this by saving all the project files into the same directory, and then archiving the directory to a file. If you do not include all necessary components, then other people will not be able to open the project.

Exporting and Printing

SmartSim can produce PNG, PDF, and SVG image files from a circuit design. You can also print out your circuit design.

Exporting

Select one of the *File – Export* options. You may be presented with a dialogue based on your choice. You will then be asked where to save the file.

PNG

You will be asked if you want a solid white background or a transparent background.

You will be asked for the resolution of the image (pixel density multiplier). A higher number increases the quality of the image, but this will increase the file size and memory usage of the image.

PDF

There are not options – you will only be asked where to save.

SVG

You will be asked if you want a solid white background or a transparent background.

Printing

If you wish to change between using a landscape and portrait layout, select *File – Page Setup*.

To print the design, select *File – Print*.

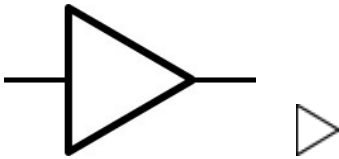
The current view in the designer window will be printed.

Built-in Component Reference

These are the basic primitive components which can be used to construct more advanced custom components and circuits.

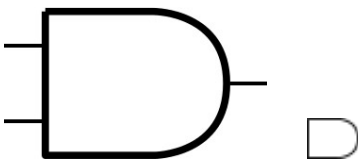
If you require an inverted form of a component, for example a NOT, NAND, NOR, or XNOR, use the Invert tool to invert the output pin of the relevant component below.

Buffer



Outputs the value of its input.

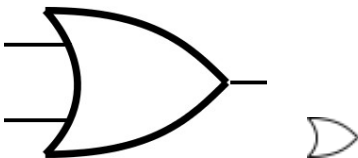
And



Outputs 1 only if all inputs are 1.

The number of inputs can be changed by using the adjust tool on the component.

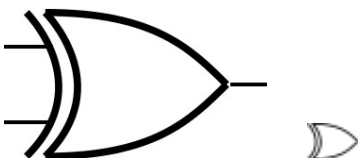
Or



Outputs 1 only if at least one input is 1.

The number of inputs can be changed by using the adjust tool on the component.

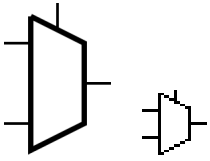
Xor



Outputs 1 only if an odd number of inputs are 1.

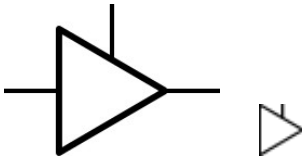
The number of inputs can be changed by using the adjust tool on the component.

Multiplexer



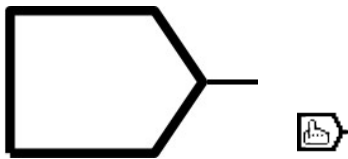
The multiplexer can be used to select an input (shown on the left).

Tristate Buffer



A tristate buffer is able to enable and disable its output to give the effect of connecting and disconnecting from the output wire. This effect is controlled by the top, perpendicular pin. The actual output, if enabled, is controlled by the left pin. The purpose of a tristate buffer is to control what part of a circuit is providing the signal on a wire, or to create a bidirectional wire or bus.

Toggle



The user can change the value a toggle outputs at circuit runtime by clicking on it. It is used to input data into a circuit.

Reader



The reader simply displays the value from its input. This is easier to read than just looking at the wire's colour.

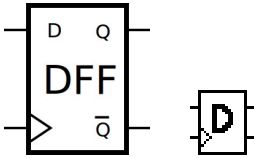
Constant



Constants are set at design time, and cannot be changed whilst the simulation is running. They constantly output the value they are set with and are used to fix a wire's state to 0 or 1.

The value can be changed by using the adjust tool on the component.

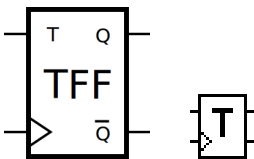
Positive-Edge-Triggered D Flip-flop



A flip-flop stores a single binary value, output as Q. In a Positive Edge Triggered D Flip-flop, the stored binary value is set to D when the clock input changes from 0 to 1.

The Q and NOT Q outputs may be left unconnected if not required.

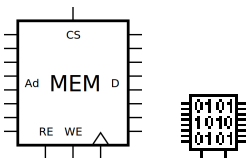
Positive-Edge-Triggered T Flip-flop



A flip-flop stores a single binary value, output as Q. In a T Flip-flop, if T is 1 then the stored binary value is toggled when the clock input changes from 0 to 1.

The Q and NOT Q outputs may be left unconnected if not required.

Memory



The memory component can store a large number of binary words. The word to access is selected using the Address pins (Ad) as a binary number. The word is accessed via the bidirectional Data pins (D) as a binary word. Chip Select (CS) is used to enable the chip. If Chip Select is 0, the chip will never read or write, and will behave as if it wasn't connected at all. If Read Enable (RE) is set to 1, the value of the word at the current address is output on the Data pins. If Write Enable (WE) is set to 1, then the word at the current address can be written to by inputting the value on the Data pins, storing the value in memory when the clock is at 1. Tristate buffers are required to control input and output with memory.

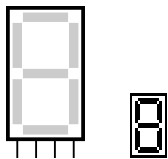
The number of Address and Data pins can be changed by using the adjust tool on the component.

You can also use the adjust tool to make it a ROM chip – this will make the memory read only. This will make the chip always output its data, as long as Chip Select is 1.

The memory component can optionally load and save its initial and final memory from and to files. This can be configured by using the adjust tool on the component.

The total number of words is set as 2^n , where n is the number of address pins. Therefore the total number of bits in a memory block is 2^nd , where d is the number of data pins. You can only emulate as much memory as your host system (the computer running the software) has free and supports. Trying to use more memory than allowed will result in a circuit compilation error.

Basic Seven Segment Display



The basic seven segment display allows you to read of up to four binary values a time, and display them as a hexadecimal number. This also allows reading decimal and octal numbers.

When displayed side-by-side with other seven segment displays in simulation, they will appear as if one component.

A “decimal point” can be added using the adjust tool.

Errors, Warnings, and Solutions

Program Start-up Errors

There was a fatal error trying to load the built in components...

This error is caused when the program cannot locate the built in component definition files.

If you used the Zip or Gzipped Tar archive installation method and there is not a “resources” folder in the same directory as the “smartsim” or “smartsim.exe” executable, this will cause the error. If you are trying to run SmartSim outside of the installation directory, this may also cause the error.

If you have compiled SmartSim using your Linux distribution's package manager, reinstalling SmartSim. If this fails, check if a later version is available.

If you have compiled SmartSim from source, you may still need to install it using the “make install-strip” command with superuser permissions.

Ensure that you have followed the installation instructions carefully – try reinstalling SmartSim.

Design Errors

Warning: Could not associate all pins with interface tags...

This is caused when you don't have interface tags with unique, consecutive IDs starting with 0.

Check your interface tags using the tag tool. You can find out which pins are mapping to what interface tags in the customiser dialogue by cycling through the pins.

Validation Errors

Circuit failed cyclic dependency test...

This is caused when a circuit contains itself. It does not need to contain itself immediately, it might for example contain itself through another component which it contains.

The error message will show a “failed ancestry”. The last component listed was the component found to be invalid, it will appear earlier in the list. The list shows how it ends up containing itself (through what components). Resolve this error by checking your circuit for design faults. A component must not contain itself.

Component has pin errors...

This is caused when pins which require a connection do not have one. Not all pins require connections. There is one pin error per invalid component.

Check that necessary component pins are bound to wires. Any offending components will be highlighted in red.

Component has unsatisfied interfaces...

This is caused when a custom component has pins which do not map to interface tags.

Check that your custom components have valid interface tagging and re-customise the component. You may need to replace each instance of the component in your circuit, by deleting and reinserting them, to use the corrected version

Circuit Warnings

Component has overlapping components...

This occurs when you have two components which have the same centre in a circuit's design.

The invalid custom component will be given in the error message. The offending components will be highlighted in red, delete them and replace with a single component.

Cannot link tagged wire in component...

This means there are more interface tags in a custom component's design than its instance within a component.

First check the customisation of your component and correct any errors. If this still fails, try replacing each instance of the component within the circuit.

Memory Chip Errors

Memory chips require that your system has enough physical memory to store all their data. If the memory requirements of your circuit are greater than the host system supports, you may either need to redesign the circuit to be less demanding, or run the circuit on a system with more memory.

Memory chips with 64 or more address wires are not supported

This current version of SmartSim does not support 2^{64} addresses. It is unlikely you should need this many addresses, and even so it would require a very, very large amount of memory.

Memory chips with X or more address wires are not supported on this host system

This is a result of trying to have more address wires than your system can support. You may be, for example, trying to use 32 or more address wires on a 32 bit system. Try using a 64 bit system if possible.

Memory chips which store 2^{64} or more bits are not supported

This version of SmartSim cannot support this much memory inside a memory chip. It is unlikely you should need this much memory.

The host system does not have enough free memory to emulate this memory chip of size X bytes

Your computer does not have enough free memory to store the memory chip's data. It may be running too many other programs at once, or not have enough memory.

The host system does not support memory chips of size greater than X bytes...

Your computer does not support memory of this size.

Simulation Errors

Multiple Output Error...

This error is a result of either a design flaw or circuit user error where two components are outputting into the same wire at the same time. The wire which has multiple signal sources will be highlighted in yellow in the simulation window.

Saving and Loading Errors

You cannot save this circuit because it contains a cyclic dependency...

This is caused when a component contains itself. It might not contain itself immediately, it might for example contain itself through another component which it contains.

The error message will show a “failed ancestry”. The last component listed was the component found to be invalid, it will appear earlier in the list. The list shows how it ends up containing itself (through what components). Resolve this error by checking your circuit for design faults. A component must not contain itself.

Failed to load due to a missing sub component...

Your custom component depends on the specified component. If the sub component is a custom component, you must load it up first. If the sub component mentioned is a built-in component, you may have an out of date version of SmartSim – check for a new version of SmartSim or try using a custom component equivalent of the built-in component.
