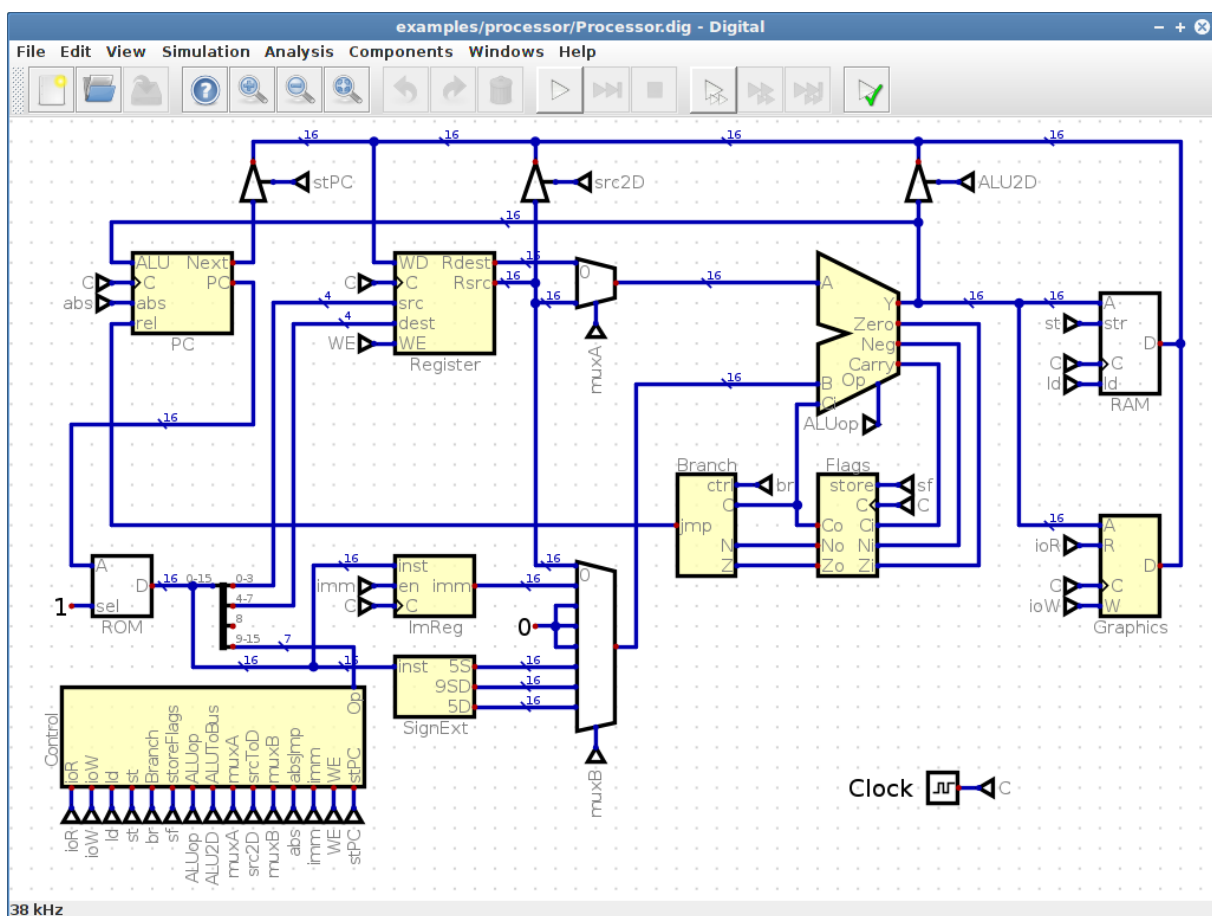


# Digital



Revision: v0.28  
Date: 2021-09-13 07:26

# Table of Contents

## A General

1. Digital .....	6
1.1. Introduction .....	6
1.2. First Steps .....	6
1.3. Wires .....	14
1.4. Hierarchical Design .....	14
2. Simulation .....	18
2.1. Propagation Delay .....	18
3. Analysis .....	18
3.1. Circuit Analysis and Synthesis .....	18
3.2. Expression .....	18
3.3. State charts .....	19
4. Hardware .....	19
4.1. GAL16v8 and GAL22v10 .....	19
4.2. ATF150xAS .....	19
4.3. Export to VHDL or Verilog .....	19
5. Custom Shapes .....	20
6. Generic Circuits .....	21
7. Script-controlled testing .....	22
8. Frequently asked Questions .....	22
9. Keyboard Shortcuts .....	24

## B Settings

## C Command Line Interface

## D Components

1. Logic	
1.1. And .....	31
1.2. NAnd .....	31
1.3. Or .....	32
1.4. NOR .....	33
1.5. XOr .....	33
1.6. XNOr .....	34
1.7. Not .....	35
1.8. Lookup Table .....	35
2. IO	
2.1. Output .....	36
2.2. LED .....	37
2.3. Input .....	37
2.4. Clock Input .....	38
2.5. Button .....	39
2.6. DIP Switch .....	39
2.7. Probe .....	40
2.8. Data Graph .....	40
2.9. Triggered Data Graph .....	41
3. IO - Displays	
3.1. RGB-LED .....	41

---

3.2. LED with two connections. ....	42
3.3. Button with LED .....	42
3.4. Seven-Segment Display .....	43
3.5. Seven-Segment Hex Display .....	44
3.6. 16-Segment Display .....	44
3.7. Light Bulb .....	45
3.8. LED-Matrix .....	45
4. IO - Mechanical	
4.1. Rotary Encoder .....	46
4.2. Stepper Motor, unipolar .....	46
4.3. Stepper Motor, bipolar .....	47
5. IO - Peripherals	
5.1. Keyboard .....	48
5.2. Terminal .....	49
5.3. Telnet .....	49
5.4. VGA Monitor .....	50
5.5. MIDI .....	51
6. Wires	
6.1. Ground .....	51
6.2. Supply voltage .....	52
6.3. Constant value .....	52
6.4. Tunnel .....	53
6.5. Splitter/Merger .....	53
6.6. Driver .....	54
6.7. Driver, inverted select .....	55
6.8. Delay .....	55
6.9. Pull-Up Resistor .....	56
6.10. Pull-Down Resistor .....	56
6.11. Not Connected .....	56
7. Plexers	
7.1. Multiplexer .....	57
7.2. Demultiplexer .....	57
7.3. Decoder .....	58
7.4. Bit Selector .....	59
7.5. Priority Encoder .....	59
8. Flip-Flops	
8.1. RS-Flip-flop .....	60
8.2. RS-Flip-flop, clocked .....	61
8.3. JK-Flip-flop .....	62
8.4. D-Flip-flop .....	63
8.5. T-Flip-Flop .....	63
8.6. JK-Flip-flop, asynchronous .....	64
8.7. D-Flip-flop, asynchronous .....	65
8.8. Monoflop .....	66
9. Memory - RAM	
9.1. RAM, separated Ports .....	67
9.2. Block-RAM, separated ports .....	68
9.3. RAM, bidirectional Port .....	69
9.4. RAM, Chip Select .....	70
9.5. Register File .....	71

9.6. RAM, Dual Port .....	72
9.7. RAM, async. ....	73
9.8. Graphic RAM .....	73
10. Memory - EEPROM .....	
10.1. EEPROM .....	74
10.2. EEPROM, separated Ports .....	75
11. Memory .....	
11.1. Register .....	76
11.2. ROM .....	77
11.3. ROM dual port .....	78
11.4. Counter .....	79
11.5. Counter with preset .....	80
11.6. Random Number Generator .....	81
12. Arithmetic .....	
12.1. Adder .....	82
12.2. Subtract .....	82
12.3. Multiply .....	83
12.4. Division .....	84
12.5. Barrel shifter .....	84
12.6. Comparator .....	85
12.7. Negation .....	86
12.8. Sign extender .....	86
12.9. Bit counter .....	87
13. Switches .....	
13.1. Switch .....	87
13.2. Double Throw Switch .....	88
13.3. Relay .....	88
13.4. Double Throw Relay .....	89
13.5. P-Channel FET .....	90
13.6. N-Channel FET .....	91
13.7. Fuse .....	91
13.8. Diode to VDD .....	92
13.9. Diode to Ground .....	92
13.10. P-Channel floating gate FET .....	93
13.11. N-Channel floating gate FET .....	94
13.12. Transmission-Gate .....	94
14. Misc. ....	
14.1. Test case .....	95
15. Misc. - Decoration .....	
15.1. Text .....	95
15.2. Rectangle .....	96
16. Misc. - Generic .....	
16.1. Generic Initialization .....	96
16.2. Code .....	97
17. Misc. - VHDL/Verilog .....	
17.1. External .....	97
17.2. External File .....	98
17.3. Pin Control .....	99
18. Misc. ....	
18.1. Power .....	99

---

18.2. Bidirectional Splitter .....	100
18.3. Reset .....	100
18.4. Break .....	101
18.5. Stop .....	101
18.6. Asynchronous Timing .....	102

## **E Library**

## A General

### 1. Digital

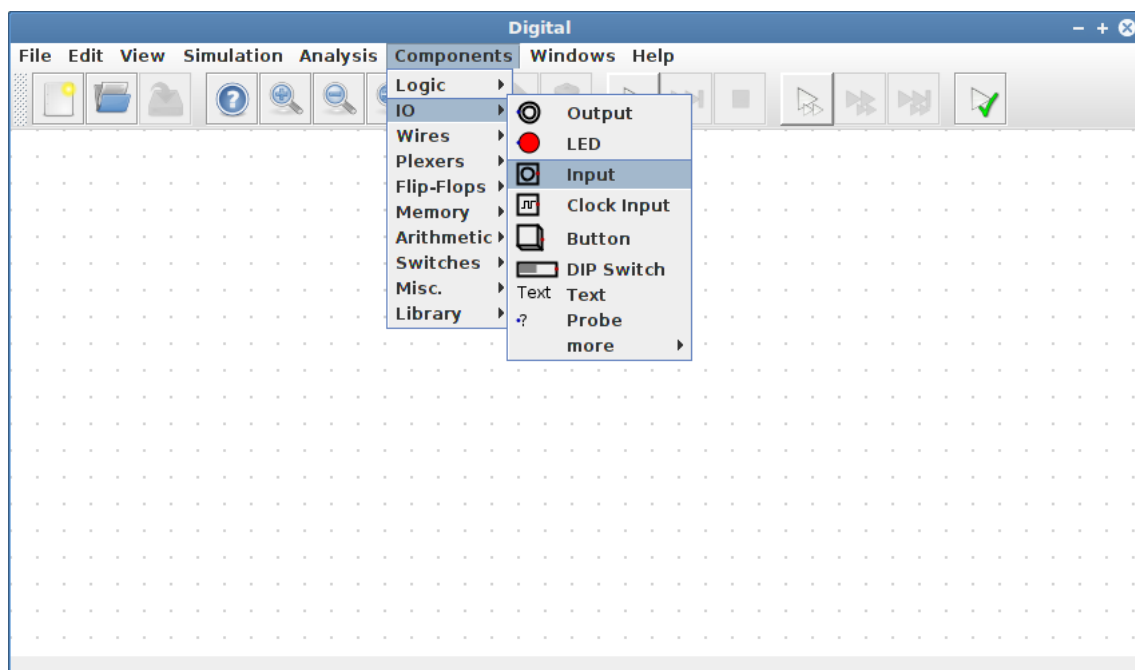
#### 1.1. Introduction

Digital is a simple simulator used to simulate digital circuits. The logic gates are connected to each other by wires and the behavior of the overall circuit can be simulated. The user can interact with the simulation by either pressing buttons or setting values to the inputs of the circuit.

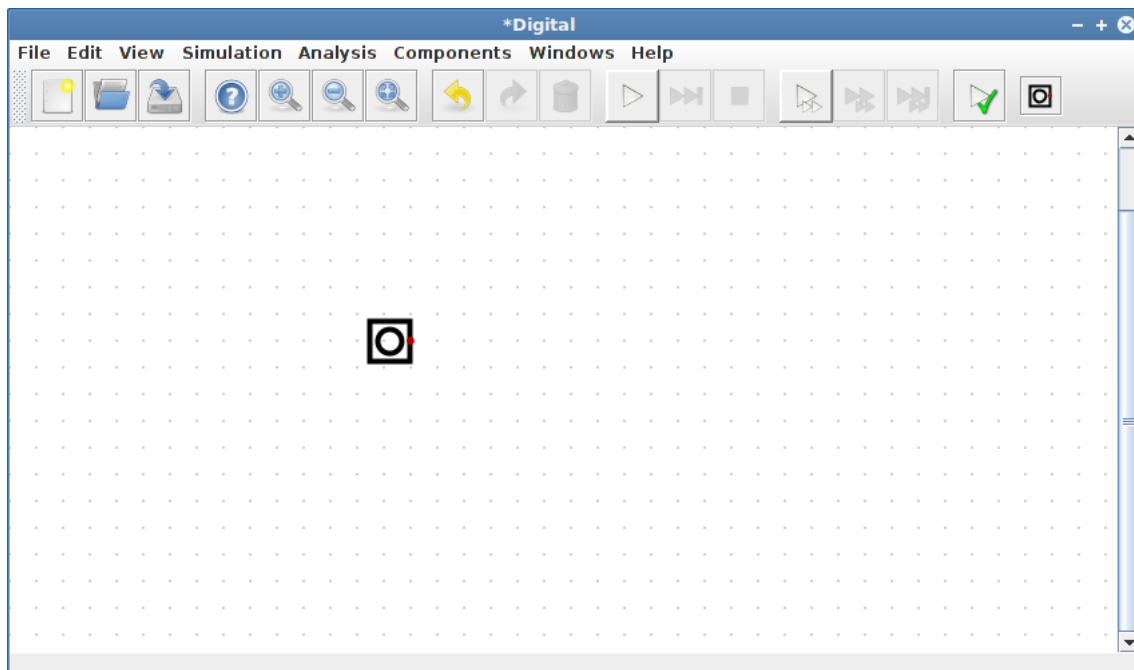
In this way, most of the basic circuits used in digital electronics can be built and simulated. In the folder *examples*, users can browse for examples that includes a functional 16-bit single-cycle Harvard processor.

The simulator has two modes of operation: Editing and Simulation mode. In the editing mode, modifications to the circuit can be performed. Users can add or connect components. In this mode, simulation is disabled. The simulation mode is activated by pressing the *Start* button in the toolbar. While starting the simulation the circuit is checked for consistency. If there are errors in the circuit an appropriate message is shown and the affected components or wires are highlighted. If the circuit is error free, the simulation is enabled. Now you can interact with the running simulation. In the simulation mode it is not possible to modify the circuit. To do so you have to activate the editing mode again by stopping the simulation.

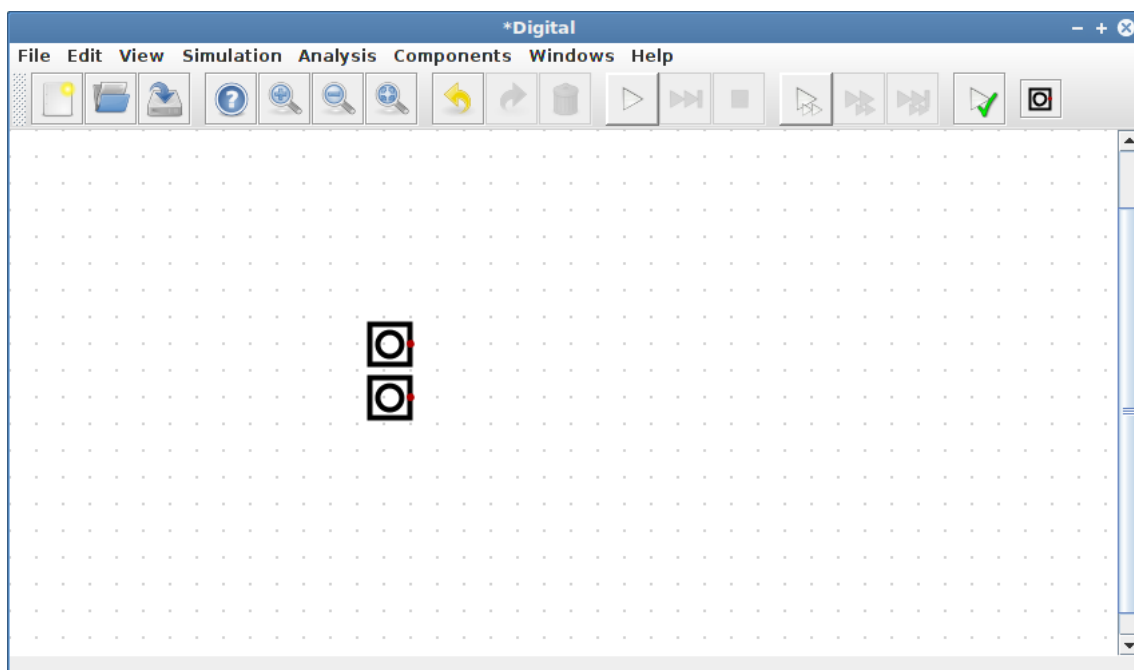
#### 1.2. First Steps



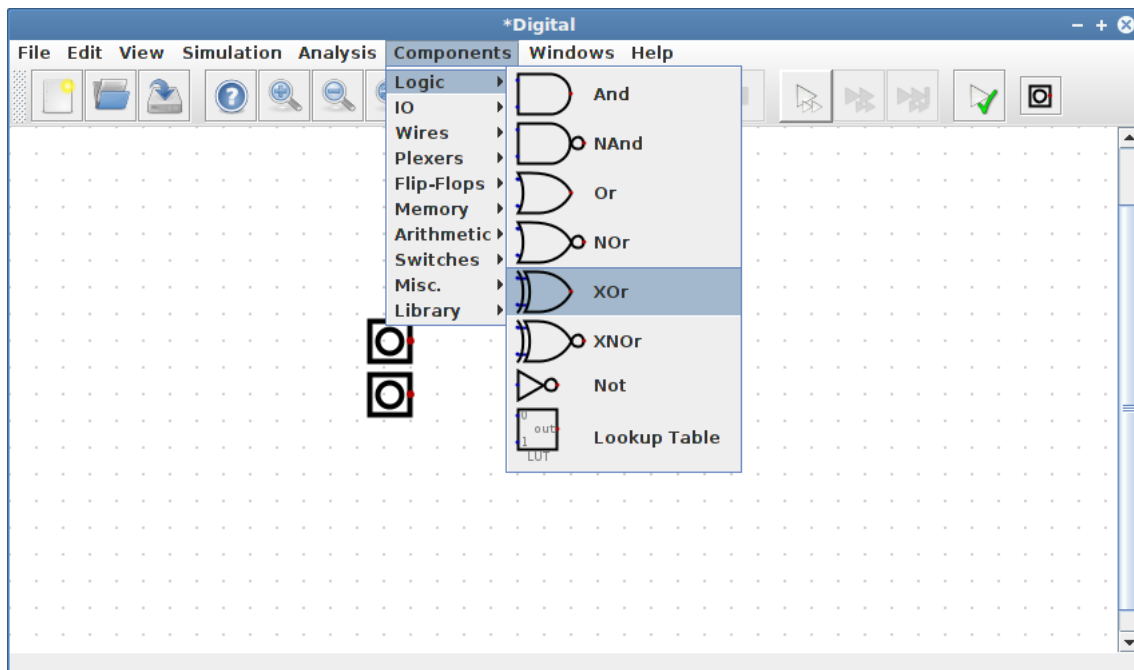
As a first example, a circuit is to be constructed with an Exclusive-Or gate. From the main window, the *Components* menu allows you to select the various components. Then they are placed on the drawing panel. This process can be canceled by pressing the ESC key at any time. Start by selecting an input component. This can later be controlled interactively by using the mouse.



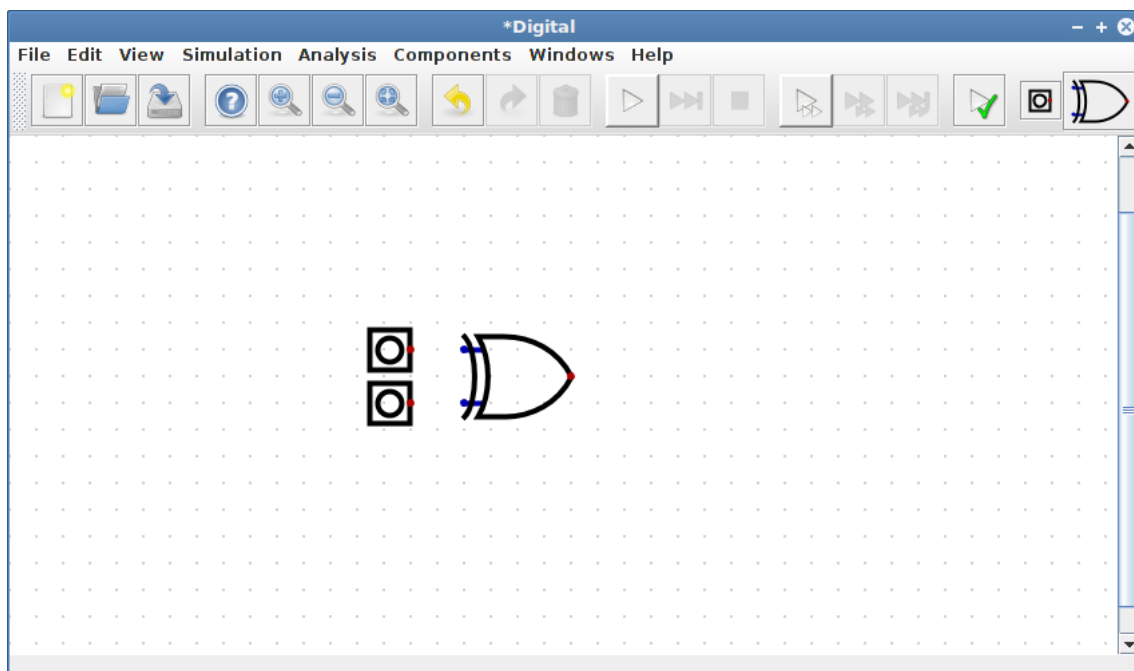
After selection, the first input can be placed on the drawing panel. The red dot on the input component symbol is a connection point between the component and a wire, which will be connected later on. The red color indicates an output. This means that the port defines a signal value or can drive a wire.



In the same way, a second input is added. It is best to place it directly below the first input.

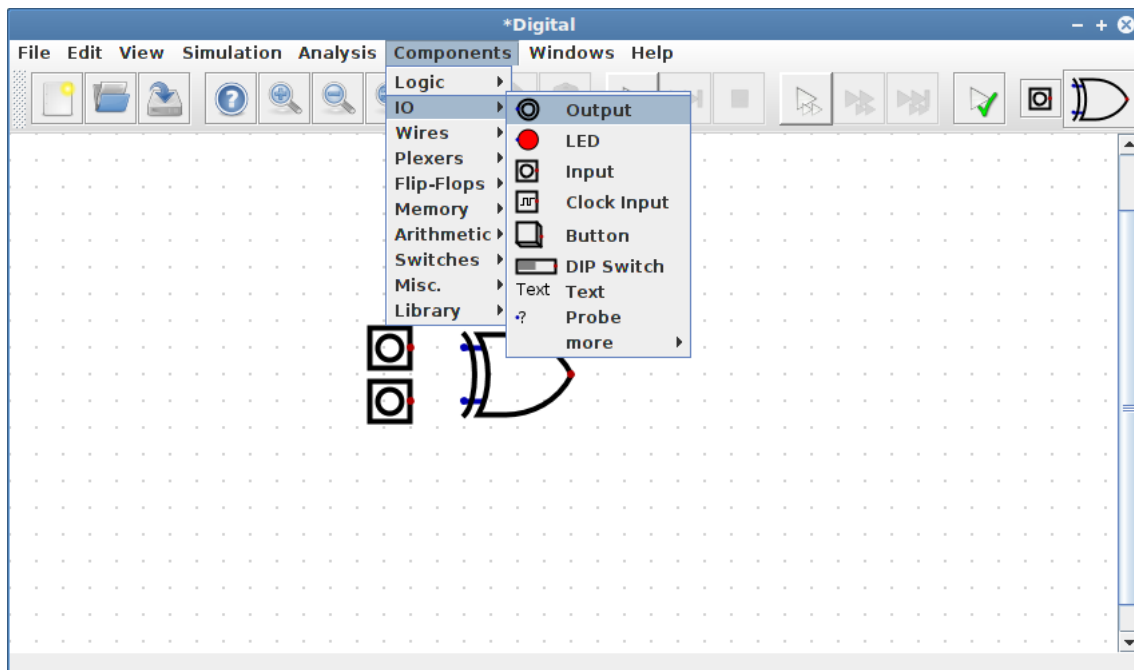


After adding the inputs, the Exclusive-Or gate is selected. This gate represents the actual logical function.

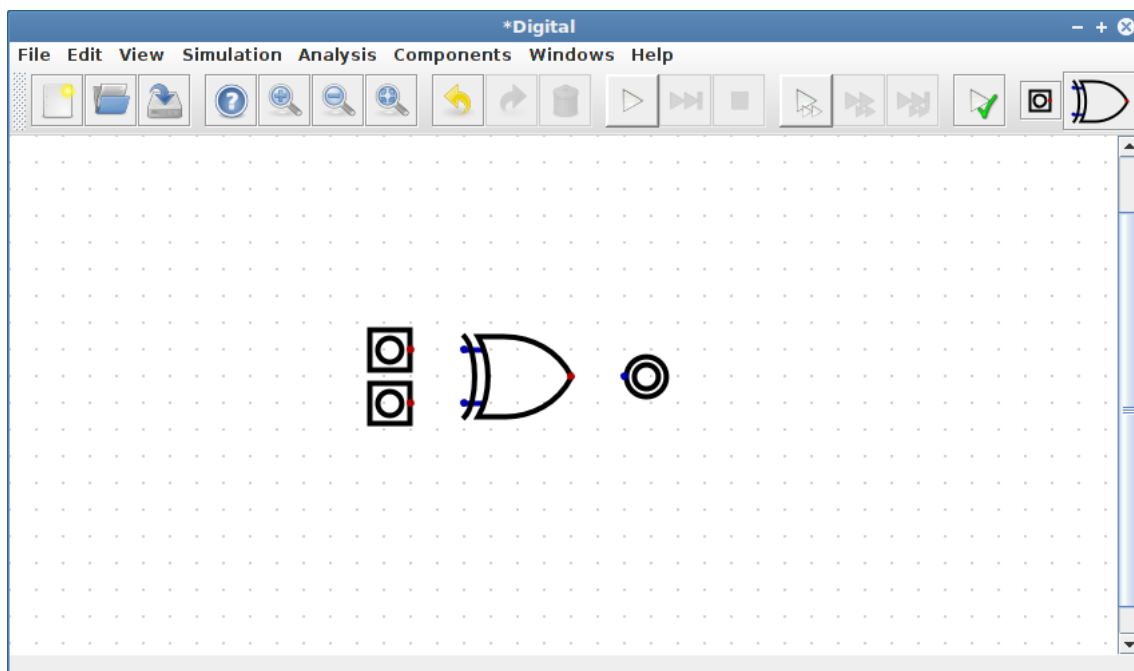


This gate can now also be added to the circuit. It is best to place it in a way that the subsequent wiring is made as simple as possible. The blue dots indicate the input terminals of the gate.

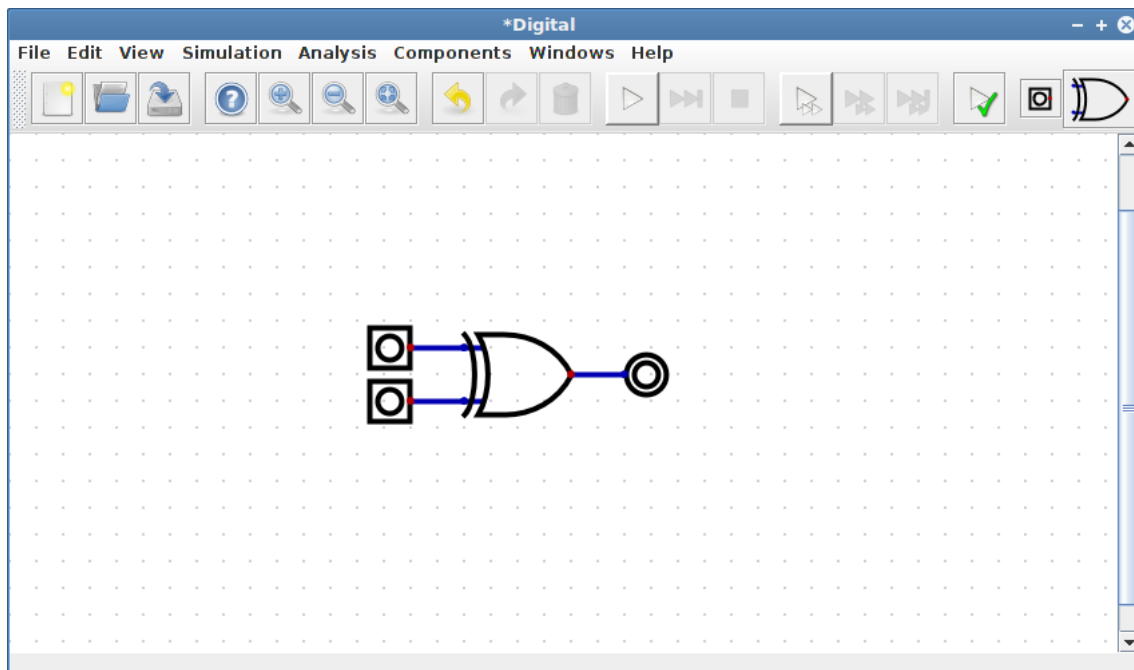




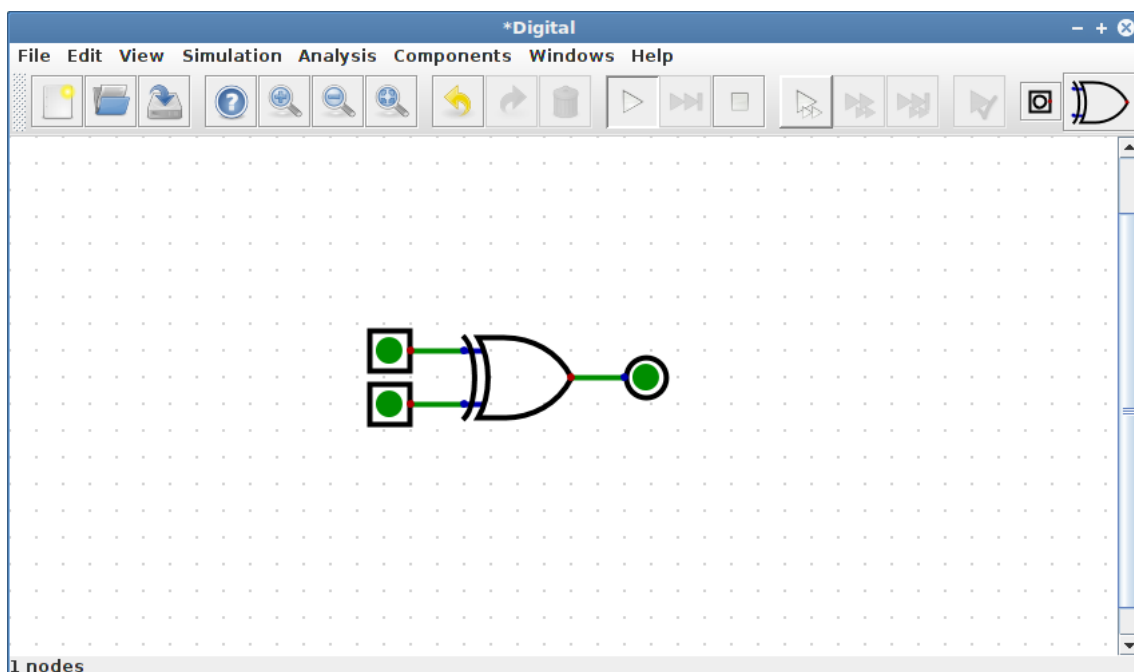
Now, select an output which could be used to display a signal state or to later pass signals to an embedding circuit.



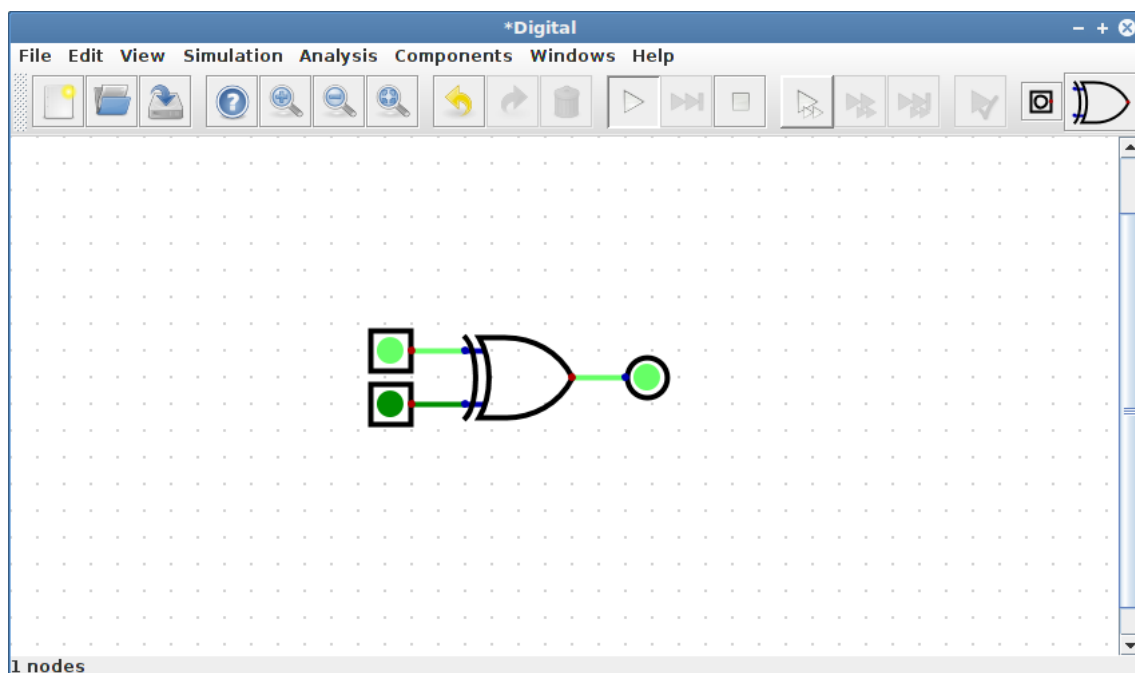
This is placed in a way that it can be wired easily. The output has a blue dot, which indicates an input terminal. Here you can feed in the value which is then exported.



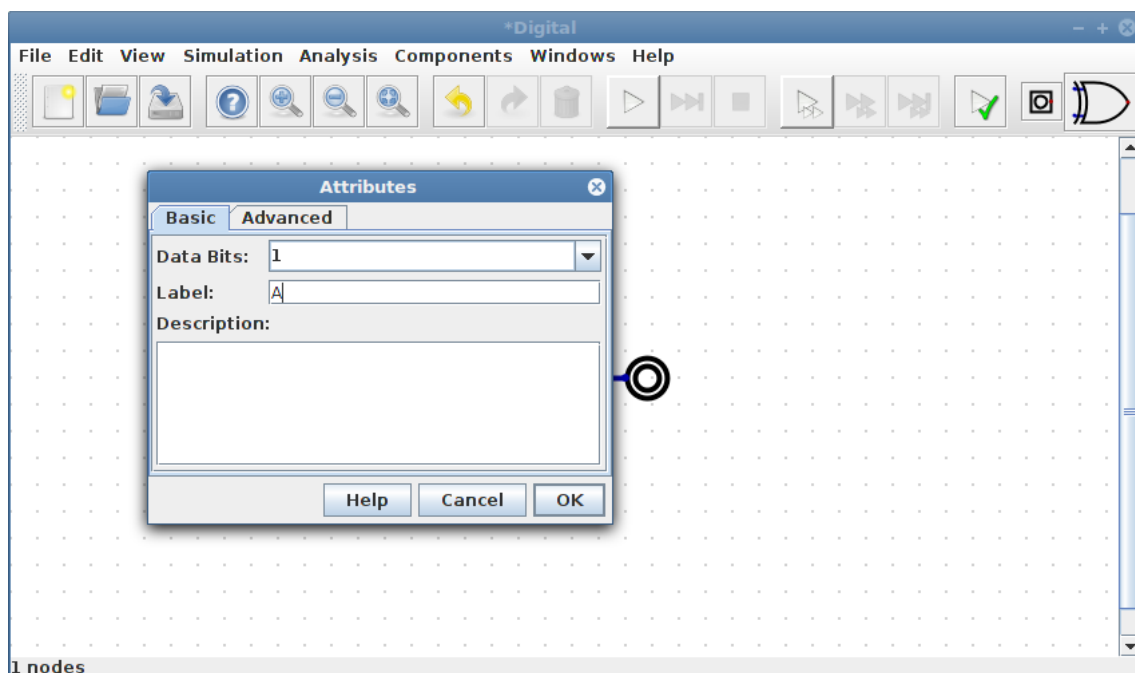
After all components are selected and in place, use the mouse to wire a connection between the blue and red dots. Make sure that exactly one red dot is connected to any number of blue dots. Only the usage of three-state outputs makes it possible to deviate from this rule and to interconnect several red dots. If all wires have been drawn, the circuit is complete.



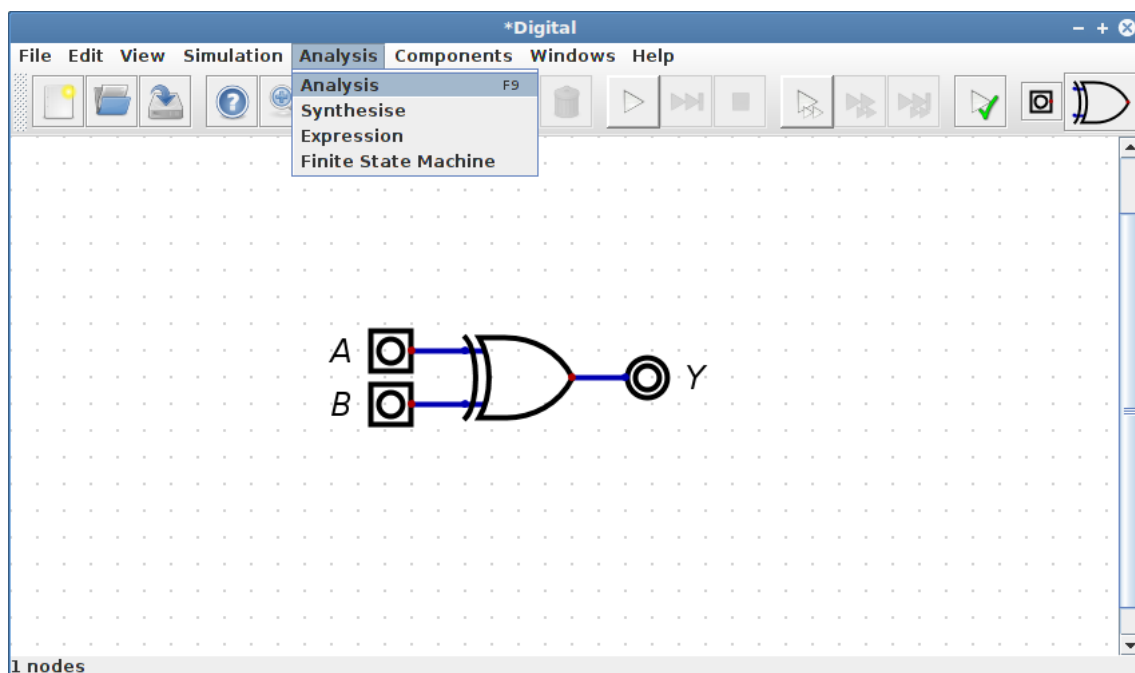
Interaction with the circuit is possible when simulation is started. This is done by clicking on the play button located in the toolbar. After starting the simulation, the color of the wires changes and the inputs and outputs are now filled. Bright green indicates a logical '1' and dark green a logical '0'. In the figure above, all wires have a '0' value.



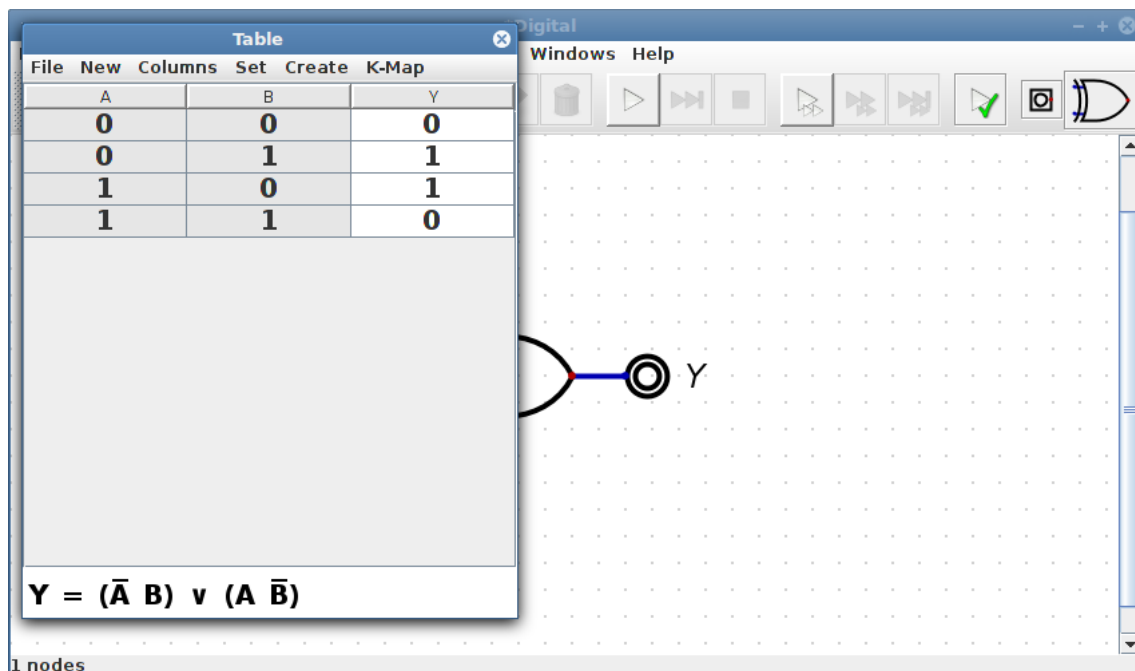
By clicking with the mouse, the inputs can be switched. Since the simulation is now active, the output changes according to the current input states. The circuit behaves like an Exclusive-Or gate as expected .



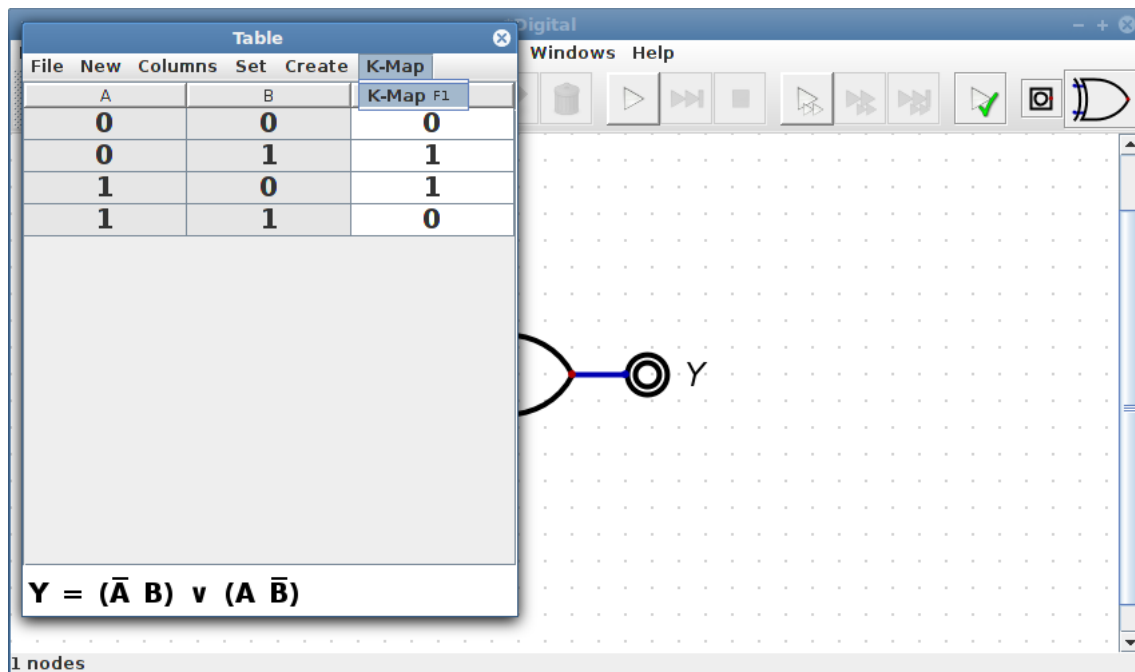
To further process the circuit, the simulation must first be stopped. The easiest way to do this is with the Stop button in the tool bar. Clicking on a component with the right mouse button (control-click on macOS) opens a dialog which shows the component's properties. The label 'A' can be defined for the first input via this dialog.



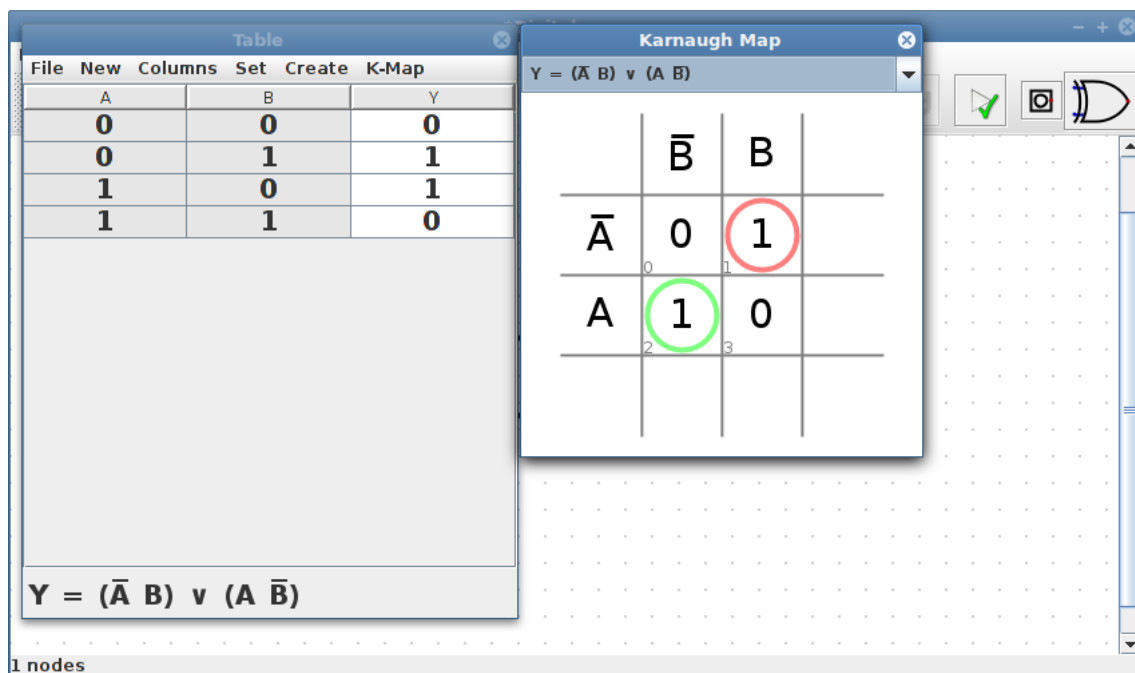
In the same way, the labels for the remaining inputs and outputs can be defined. The menu item *Analysis* also contains a menu item *Analysis*. This function performs an analysis of the current circuit. However, this is only possible if all inputs and outputs are labeled properly.



The truth table of the simulated circuit appears in a new window. Below the table you can find the algebraic expression associated with the circuit. If there are several possible algebraic expressions, a separate window will open, showing all possible expressions.



The table dialog has the menu entry *K-Map* in its main menu. This allows to display the truth table in the form of a K-map.



At the top of this dialog there is a drop-down list which allows the selection of the desired expression in the K-map. In this way you can, for example, illustrate how several equivalent algebraic expressions can result. However, in this example, there is only one minimal expression. The truth table can also be modified by clicking the K-map.

### 1.3. Wires

All components must be connected via wires. It is not possible to connect two components by placing them directly next to each other.

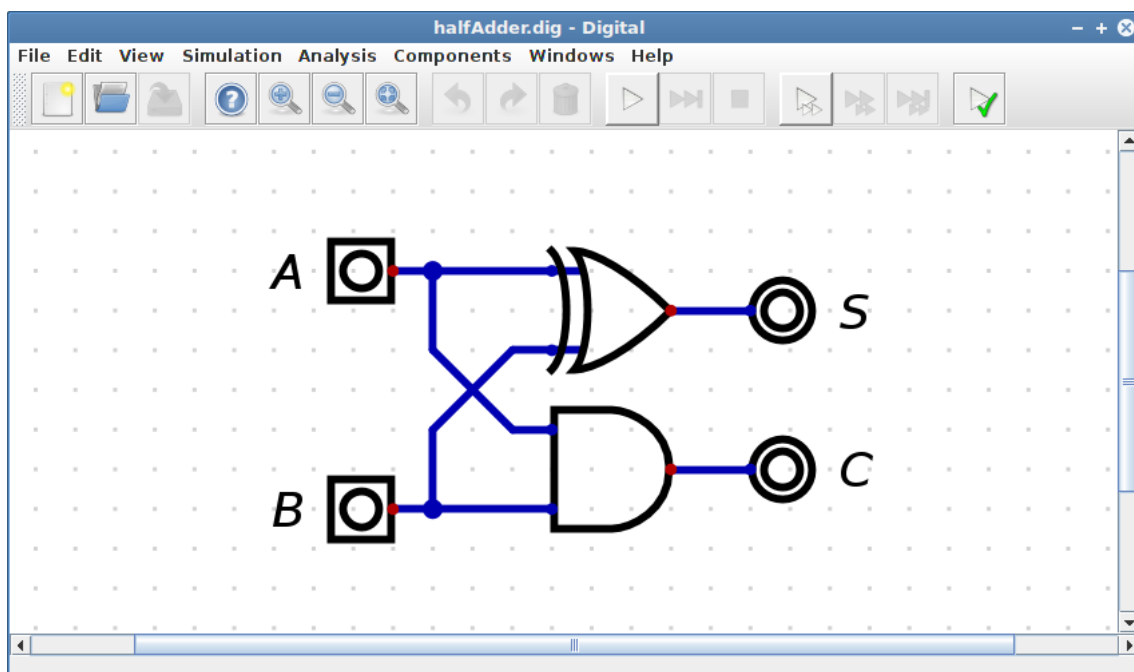
In addition, there are only connections between an endpoint of a wire and a component. If a pin of a component is placed in the middle of a wire, no connection is made between the component and the wire. Therefore, a wire must actually terminate at each pin which is to be connected. Even if the tunnel component is used, there must be a wire between the pin and the tunnel element.

The component needs to be selected using the rectangular selection tool in order to be moved along with the connected wires. For moving a component without the connected wires, select the component using a mouse click.

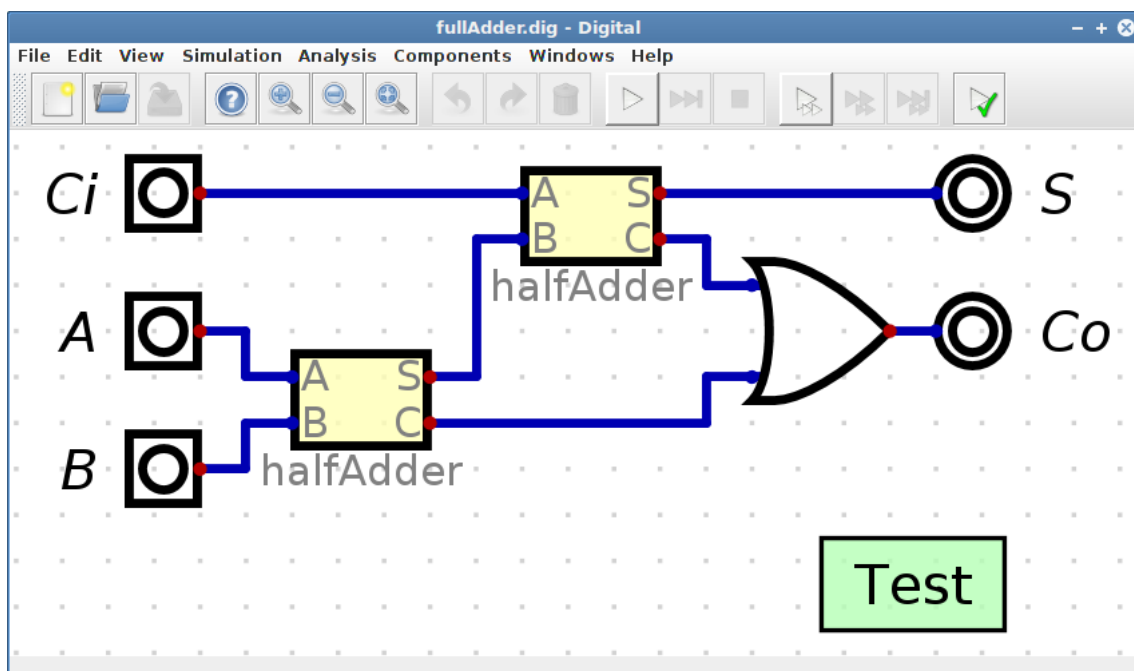
With CTRL-Click a single wire section can be selected to move or delete it. If the D key is pressed while drawing a wire, a diagonal wire can be drawn. The key S allows the splitting of a line segment into two segments.

### 1.4. Hierarchical Design

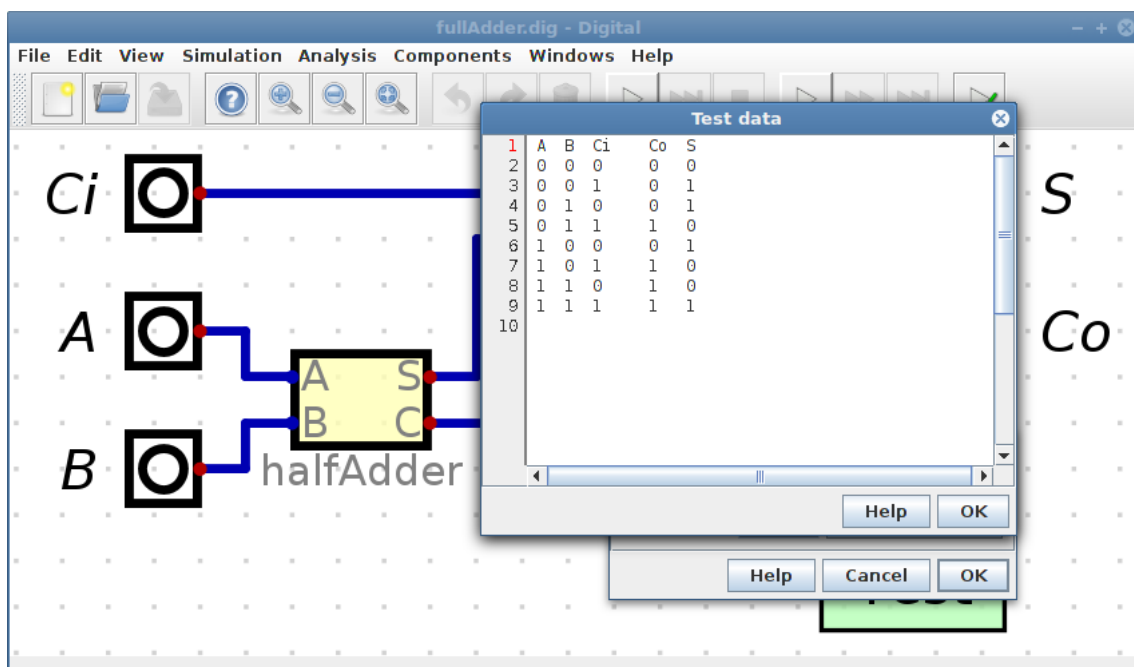
If a complex circuit is built up, this can quickly become very confusing. To keep track here, the different parts of a circuit can be stored in different files. This mechanism also makes it possible to use a subcircuit, which has been created once, several times in a further circuit. This approach also offers the advantage that the files can be stored independently of each other in a version control system and changes can be tracked.



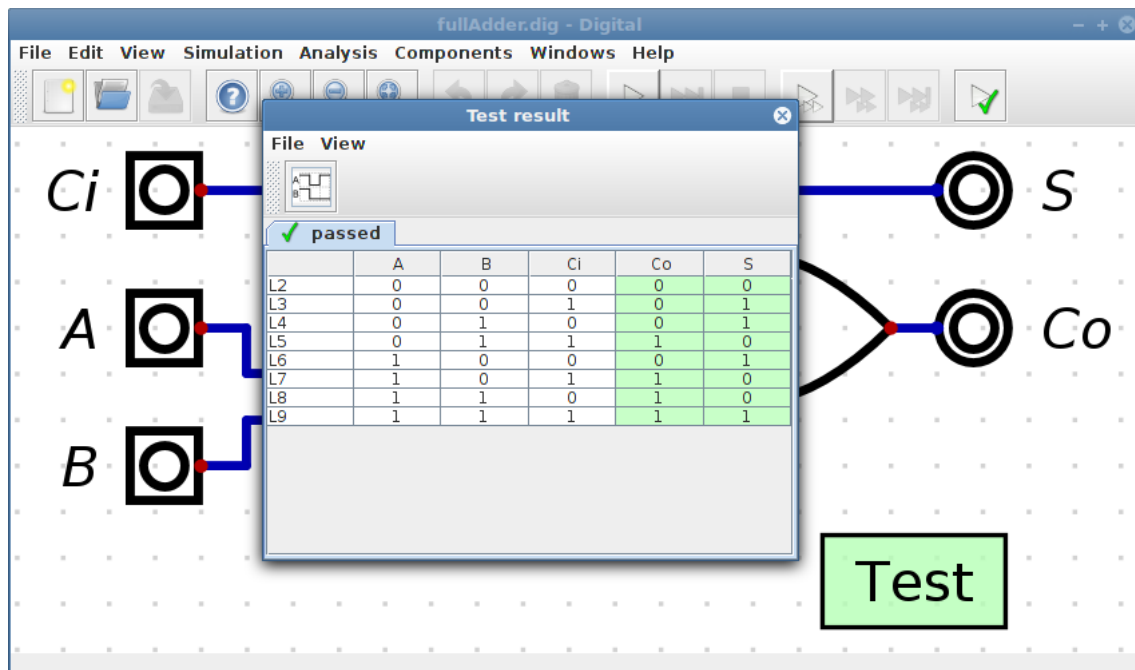
As an example, consider a 4-bit adder: First, we built a simple half-adder. This consists of an XOR gate and an AND gate. The sum of the two bits 'A' and 'B' is given to the outputs 'S' and 'C'. This circuit is stored in the file *halfAdder.dig*.



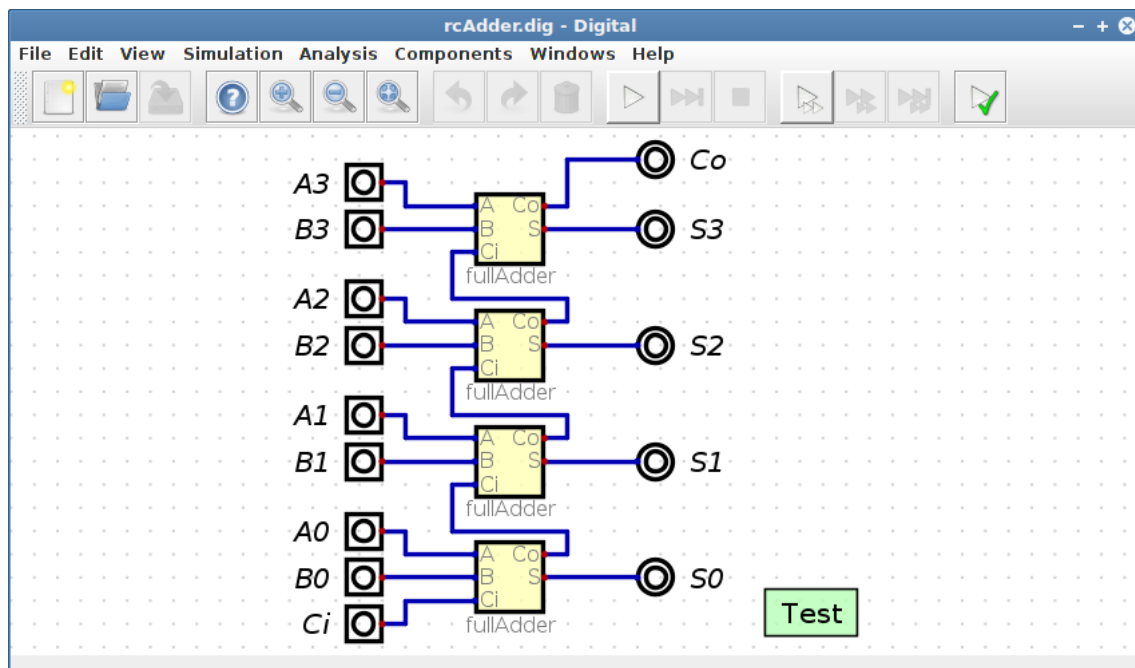
From two half adders a full adder can now be built. To do this, create a new empty file and save the empty file as *fullAdder.dig* in the same folder as the half adder. Then the half adder can be added to the new circuit via the *Components* → *Custom* menu. The order of the pins at the package of the half-adder can be rearranged from the half adder in the menu *Edit* → *Order inputs* or *Edit* → *Order outputs*. The full adder adds the three bits 'A', 'B' and 'Ci' and gives the sum to the outputs 'S' and 'Co'.



In order to check the correct function of the full adder, a test case should be added. In the test case, the truth table is stored, which should fulfill the circuit. In this way it can be automatically checked whether this is the case.

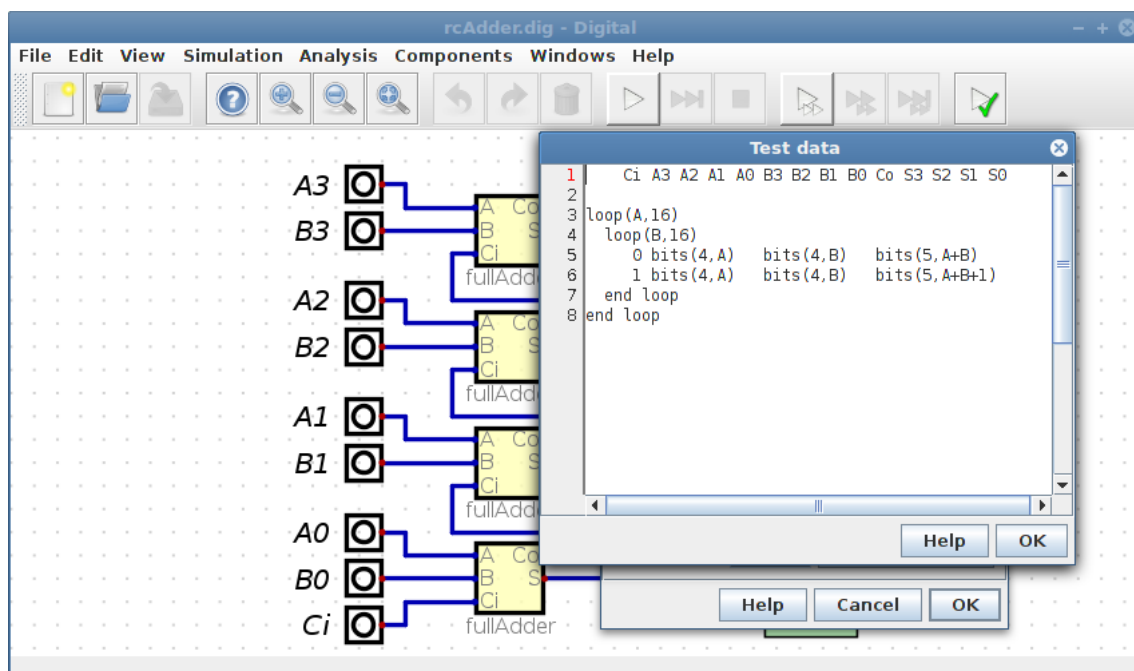


The tests can be executed via the test case editor or the test button in the toolbar. The table cells highlighted in green indicate that the output of the circuit matches the truth table given in the test case.



Now the full adders can be put together to form a so-called ripple-carry adder. In this case, the carry output of an addition is forwarded as a carry input to the addition of the next higher-order bit, just as is usual in pencil-and-paper addition. This 4-bit adder should be tested for correct function. For this purpose a test case was inserted.





This test case performs a 100% test, which is possible only with relatively simple circuits: all possible 512 input combinations are applied to the circuit, and it is checked whether the output of the circuit is correct. The first line lists the input and output signals. Below this, the input values to be applied and the output values to be checked are specified in a row, as in a truth table. In this example, however, 512 lines are required. Entering this would be a tedious and error-prone task. It is easier and more reliable to automatically generate the required lines. For this purpose, the variables *A* and *B* are each traversed from 0 to 15. The respective values of *A* and *B* are then assigned to inputs '*A*[*n*]' and '*B*[*n*]'. Then it is checked whether the circuit outputs the value *A*+*B*. Then it is checked again with the carry bit set, in which case *A*+*B*+1 must result. The details of the test syntax are provided by the help dialog.

If a circuit is embedded in an other circuit, only the file name of the subcircuit is stored in a circuit, not the embedded circuit itself. The corresponding files of the embedded subcircuits must therefore be found in the file system at runtime of the simulation. In order to support the various work methods of the users as best as possible and still to avoid a complex administration of import paths, etc., a somewhat unusual import strategy is implemented.

Only the file names of the embedded circuits are stored in a circuits file, not the full path. If a file needs to be opened, all subfolders are searched for a file of the corresponding name. If a suitable file is found, it is imported. This process only depends on the file name of the file to be read, not on its path. Correspondingly, an error message is generated if there are several files of the same name in different subfolders, since ambiguities then arise.

A suitable project structure therefore looks as follows: The root circuit is located in a separate folder. All imported circuits must be in the same folder or subfolders. All circuits must have different names, so it must not happen that there are circuits of the same name in different folders.

## 2. Simulation

### 2.1. Propagation Delay

During the simulation every logic gate has a propagation delay. Every component found in the library has the same propagation delay regardless of its complexity. The AND gate thus has the same propagation delay as the multiplier. The only exceptions are diodes, switches and splitters which are used to create data buses. These components have no propagation delay at all.

If it's necessary to simulate a gate - e.g. the multiplier - with a longer propagation delay, a delay gate must be inserted in the circuit right behind the output of the multiplier.

If a circuit is included in another parent circuit, the included circuit keeps its timing behaviour. So if you include a complex circuit which has a large propagation delay because the input signals has to pass three gates until it reaches the output, this behaviour is conserved while including this circuit. There are no additional delays introduced as a result of including a circuit. If not all outputs of a circuit have the same propagation delay, then this is also the case if it is included in a parent circuit. In general, including a circuit into an other circuit does not modify its timing behaviour at all. An included circuit behaves exactly the same way as if all components had been inserted at the same circuit level.

## 3. Analysis

### 3.1. Circuit Analysis and Synthesis

A circuit can be analyzed via the menu entry *Analysis*. A truth table is generated for purely combinatorial circuits. This truth table can be edited as desired. A new circuit can be generated from this truth table after editing.

In addition to purely combinatorial circuits, it is also possible to analyze or generate sequential circuits. Instead of a simple truth table a so-called state transition table is created. Each flip-flop thereby appears on the input side and the output side of the state transition table. In this table, on the right-hand side, you can find the next state, which will occur after the next clock signal. This next state depends on the current state of the flip-flops as found at the left-hand side of the table. For an analysis to be possible, the flip-flops must be named.

The following naming convention applies: The following next state of a bit on the right side of the table is indicated by a lowercase 'n+1'. The corresponding current state is indicated by an appended 'n'. If there is a state variable 'A', 'An' indicates the current state and 'An+1' indicates the next state. If, in the truth table on the left and right side, signals are present, which correspond to this pattern it is assumed that the table is a state transition table, and a sequential circuit is generated instead of a combinatorial circuit.

It should be noted that the circuit to be analyzed may contain only purely combinatorial elements in addition to the built-in D and JK flip-flops. If a flip-flop is e.g. made from Nor gates, this circuit is not recognized as a flip-flop and therefore it is not possible to analyse such a circuit.

### 3.2. Expression

Via the menu item *Expression* it is possible to enter a boolean function from which a circuit can then be generated.

### 3.3. State charts

An editor for state charts is available via the menu item *Finite State Machine*. It allows the graphical creation of state machines by drawing states and state transitions. Thereby different outputs can be set in the different states. By providing transitions with conditions, input signals can be generated. By setting output values at transitions, Mealy automata can also be defined. The state machine defined in this way can then be automatically transferred to a state transition table, from which, in a further step, a circuit implementing the initial state machine can be generated. If the simulation of this circuit is then started, the current state can also be followed in the state chart.

## 4. Hardware

### 4.1. GAL16v8 and GAL22v10

In the circuit generation menu in the truth table there are also functions to generate so-called JEDEC files. This is a special file format that describes the fuse map of a PLD. This JEDEC file can be written into a corresponding PLD using a special programmer. At the moment, circuits of the type *GAL 16v8* and *GAL22v10* or fuse map compatible devices are supported.

### 4.2. ATF150xAS

The chips in the *ATF150x* family are simple CPLDs with up to 128 macrocells. They are available in a PLCC package, which makes them suitable for laboratory exercises: If an IC is destroyed during exercises, it can simply be replaced. In addition, with the *ATDH1150USB* an easy to use, low-cost programmer is available. This programmer is able to program the *ATF150x* chips in system using a JTAG interface. A suitable evaluation board (*ATF15XX-DK3-U*) is also available. The software *ATMISP*, which is available on the ATMEL/Microchip website, is required for programming the chips.

Unfortunately, the fuse map details are not publicly available so that no suitable fitter for this chip can be integrated in Digital, as is possible with the *GAL 16v8* and *GAL22v10* chips.

Therefore, the fitters *fit150[x].exe* provided by ATMEL must be used. These programs create a *JEDEC* file from a suitable *TT2* file which can then be programmed on the chip. Digital starts the fitter automatically every time a *TT2* file is created. For this purpose, the path to the *fit150[n].exe* fitters must be specified in the settings. The created *JEDEC* file can then be opened and programmed directly with *ATMISP*.

For legal reasons the fitter *fit1502.exe* can not be distributed with Digital. However, it can be found in the folder *WinCupl\Fitters* after installing *WinCupl*. *WinCupl* is available on the ATMEL/Microchip website. On Linux systems, the fitters can also be executed by Digital if *wine* is installed.

### 4.3. Export to VHDL or Verilog

A circuit can be exported to VHDL or Verilog. A file is generated which contains the complete description of the circuit. The generated VHDL code was tested with Xilinx Vivado and the open source VHDL simulator ghdl. The Verilog code is tested with the Verilog simulator Icarus Verilog.

If a circuit contains test cases, the test data is used to generate a HDL test bench. This can be used to check the correct function of the circuit in a HDL simulation.

Additional files which are needed by special boards can be created. At present only the BASYS3 board and the Mimas boards Mimas and Mimas V2 are supported. A constraints file is created, which contains the assignment of the pins. The description of the pins can be found in the boards data sheet, and must be entered as a pin number for the inputs and outputs.

For a BASYS3 board, if the circuit clock frequency is low, a frequency divider is integrated into the HDL code to divide the board clock accordingly. If the clock frequency selected in the circuit exceeds 4.7MHz, the MMCM unit of the Artix-7 is used for clock generation. This ensures that the FPGA resources provided for the clock distribution are used. This allows the included example processor to run at 20MHz, and if you can do without the multiplier, 30MHz is also possible.

If a circuit is to run on a BASYS3 board, a new project can be created in Vivado. The generated VHDL file and the constraints file must be added to the project. Once the project has been created, the bitstream can be generated and the Hardware-Manager can be used to program a BASYS3 board.

In order to create the required constraints file in addition to the HDL file, the corresponding board must be configured in the settings. In the field "Toolchain Configuration" the corresponding XML file can be selected. The available configurations can be found in the folder *examples/hdl* and have the file extension *.config*. If the configuration was successfully integrated, a further menu appears, which makes the board specific functions accessible.

## 5. Custom Shapes

Although Digital has some options that determine the appearance of a circuit when it is embedded in another, in some cases it may be useful to use a very special shape for a subcircuit. An example is the representation of the ALU in the processor included in the examples. This chapter explains how to define such a special shape for a circuit.

Digital does not provide an editor for creating a special shape. Instead, a small detour is required for creating circuit shapes: First, the circuit which is to be represented by a special shape is opened. Then an SVG template is created for this circuit. In this template, the circuit is represented by a simple rectangle. It also contains all the pins of the circuit, represented by blue (inputs) and red (outputs) circles. To see which circle belongs to which pin, you can look at the ID of the circle in the object properties. This ID has the form *pin:[name]* or *pin+:[name]*. In the latter variant, the pin is provided with a label if reimported to digital. If you do not want such a label, the + can be removed.

This SVG file can now be edited. The most suitable is the open source program Inkscape which is available for free. The pins can be moved freely, but are moved to the next grid point during the reimport.

If existing SVG files are to be used, it is easiest to open the created template and paste the existing graphic into the template via Copy&Paste.

If the file was saved, it can be imported with Digital. The file is read in and all necessary information is extracted and stored in the circuit. For further use of the circuit, the SVG file is no longer required.

A final remark: SVG is a very powerful and flexible file format. It can be used to describe extremely complex graphics. The Digital importer is not able to import all possible SVG files without errors. If a file can not be imported, or does not appear as expected, some experimentation may be required before the desired result is achieved.

## 6. Generic Circuits

It happens that a subcircuit has been created, and is to be used in different variants. For example, you can imagine a special counter that is needed for different bit widths. If one would create a subcircuit for 4, 5 and 6 bits each, the maintenance of the circuit would be difficult in the future, since one must always work on several subcircuits, which are identical except for one parameter, the bit width.

To prevent this, a generic subcircuit which can be parameterized can be created. For this purpose, the checkbox "Circuit is generic" must be set in the circuit specific settings. Then the parameter dialog of each component in that circuit contains the additional field "generic parameterization". In this field program code can be entered, which can change the parameters of the component. Each parameter has a name and can be modified as an attribute of the field *this*. The names of the parameters can be found in the help dialog of the component. If you want to change the bit width of an adder, the line *this.Bits=1;* can be used.

In this way, however, it is not yet possible to create a circuit that can be parameterized. It is still necessary to access parameters that are set when the circuit is used. This is done via the field "args". If you want to set the bit width from outside, you can write: *this.Bits=args.bitWidth;*. The name of the argument - here *bitWidth* is arbitrary. If such a subcircuit is used, this argument must be set.

If the circuit is used and the parameter dialog of the embedded circuit is opened, it also has a field "generic parameterization". Here the bit width to be used can be set with the instruction *bitWidth:=5;*.

If a generic circuit is to be started directly, this is not possible straight away, since the required arguments are missing, which have to be specified when embedding the circuit. These missing arguments would lead to corresponding error messages. Therefore, to simplify the testing of the circuit, the *Generic Initialization* component can be added to the circuit. In this component you can set the arguments that would come from an embedding circuit. In this way, a generic circuit can also be simulated directly. If the circuit is embedded, this component is ignored. It is only needed for the direct start of the simulation.

Under certain circumstances it may be useful not only to change the attributes of the components of a circuit, but to add completely new components and wires depending on the passed arguments. The *Code* component can be used for this purpose. If it is added to the circuit, the contained Code will be executed when the simulation is started. Here, a wire can be added using the *addWire([x1],[y1],[x2],[y2])* function, and using the function *addComponent([name],[x],[y])* a new component *[name]* can be added at the position *([x],[y])*. The return value of the *addComponent([Name],[x],[y])* function allows to set the parameters of the component.

The example circuit *examples/generic/modify/Conway/GenericConway.dig* shows how a more complex circuit can be assembled in this way.

Another way to create a circuit is recursion: it is possible, depending on the arguments, to replace one circuit by another. For this purpose the function *setCircuit([Name])* is available. If it is called in the definition part of a subcircuit, the circuit to be inserted can be replaced by another circuit. This allows the recursive definition of a circuit. As in other programming languages, a suitable termination condition must be ensured.

The *examples/generic* folder contains an example of a Gray code counter whose bit width can be configured. Here a Gray code counter is constructed by recursively adding further bits to an initial circuit until the required number of bits of the counter is reached.

## 7. Script-controlled testing

If students are to complete exercises with Digital, it can be helpful if the circuits submitted by the students can be checked in an automatic process. To perform this check, Digital can be started from the command line. The call is done as follows:

```
java -cp Digital.jar CLI test [file to test] [-tests [optional file with test cases]]
```

If only the file to be tested is specified, the test cases in that file are executed. In this way, the test cases created by the students themselves can be executed.

If a second file name is specified, the test cases are taken from the second file and the first circuit is checked with these test cases. The second file will therefore usually contain the sample solution whose test cases are complete and correct. The circuit contained in the second file is ignored. Only the test cases are taken from it.

In order to test a submitted circuit against a sample solution, the signal names of the inputs and outputs in both circuits must match.

## 8. Frequently asked Questions

### How to move a wire?

Select one of the end points with the rectangular selection. Then move this point using the mouse. You can also select a wire with CTRL + mouse button.

### How to delete a wire?

Select one of the end points and press *DEL* or click on the trashcan. You can also select a wire with CTRL + mouse button.

### How to move a component including all the connected wires?

Select the component with the rectangular selection. The selection must include the entire component. Then move the component including the wires using the mouse.

### There is a component not connected to a wire, even though the pins are on the wire.

A pin is only connected to a wire if the wire has an endpoint at the pin.

### If the names of the pins in a circuit are long, the names are no longer readable when the circuit is embedded. What can I do?

The width of the block can be increased using the menu item *Edit* → *Circuit specific settings*.

### The pins in an embedded circuit have an non-optimal order. How can this be changed?

The sequence can be changed using the menu entry *Edit* → *Order inputs* or *Edit* → *Order outputs*.

### When the simulation is started, a wire becomes gray. What does that mean?

The colors light green and dark green are used to represent high and low state. Gray means the wire is in high Z state.

**I have a truth table. How to calculate the minimized boolean equations?**

In the menu *Analysis* select the entry *Synthesise*. Then enter the truth table. At the bottom of the window you can find the matching boolean equation. If you enter more than one dependent variable, a new window opens in which all boolean equations are shown.

**I have entered a truth table, but there is more than one boolean equation shown. Which of them is the correct one?**

Minimizing a boolean equation can result in many equations describing the same function. Digital shows all of them and they all create the same truth table. There may be differences depending on the "don't cares" in the truth table.

**I have a truth table. How to create a circuit representing the truth table?**

In the menu *Analysis* select the entry *Synthesise*. Then enter the truth table. You can edit the table using the *New* or *Edit* menus. In the menu *Create*, you can create a circuit using the *Circuit* item.

**How to edit a signal's name in the truth table?**

Right click on the name in the table header to edit the name.

**I have a boolean equation. How to create a circuit?**

In the menu *Analysis* select the entry *Expression*. Then enter the equation.

**How to create a truth table from a boolean equation?**

In the menu *Analysis* select the entry *Expression*. Then enter the expression. Then create a circuit and in the menu *Analysis* use the entry *Analysis* to create the truth table.

**How to create a JEDEC file from a given circuit?**

In the menu *Analysis* select the entry *Analysis*. Then in the menu *Create* in the new window choose the correct device in the sub menu *Device*.

**When creating a JEDEC file: How to assign a pin number to a certain signal?**

At the corresponding inputs and outputs you can enter a pin number in the settings dialog of the pin.

**I have created a JEDEC file. How to program it to a GAL16v8 or GAL22v10?**

To program such a chip a special programmer hardware is necessary.

**I have created a circuit that I want to use in many other circuits. How can I do this without copying the file over and over again into the appropriate folders?**

The circuit can be saved in the "lib" folder. Then it is available in all other circuits.

## 9. Keyboard Shortcuts

<b>Space</b>	Starts or stops the simulation.
<b>F6</b>	Opens the measurement table dialog.
<b>F7</b>	Run to Break.
<b>F8</b>	Execute test cases.
<b>C</b>	A single clock step (Works only in a running simulation and only if there is a single clock component).
<b>V</b>	Execute a single gate step.
<b>B</b>	Execute all single gate steps until the circuit has stabilized or, if a break component is present, until the break.
<b>F9</b>	Analysis of the circuit.
<b>CTRL-A</b>	Select all.
<b>CTRL-X</b>	Cuts the selected components to the clipboard.
<b>CTRL-C</b>	Copys the selected components to the clipboard.
<b>CTRL-V</b>	Inserts the components from the clipboard.
<b>CTRL-D</b>	Duplicate the current selection without modifying the clipboard.
<b>R</b>	While inserting this rotates the components.
<b>L</b>	Inserts the last inserted component again.
<b>T</b>	Inserts a new tunnel.
<b>CTRL-N</b>	New circuit.
<b>CTRL-O</b>	Open circuit.
<b>CTRL-S</b>	Save the circuit.
<b>CTRL-Z</b>	Undo last modification.
<b>CTRL-Y</b>	Redo the last undone modification.
<b>P</b>	Programs a diode or a FGFET.
<b>D</b>	While drawing a wire switches to the diagonal mode.
<b>F</b>	While drawing a line flips the orientation.
<b>S</b>	Splits a single wire into two wires.
<b>ESC</b>	Abort the current action.
<b>Del</b>	Removes the selected components.
<b>Backspace</b>	Removes the selected components.
<b>+</b>	Increases the number of inputs at the component the mouse points to. If it is used with constants, the value is increased.
<b>-</b>	Decreases the number of inputs at the component the mouse points to. If it is used with constants, the value is decreased.



<b>CTRL +</b>	Zoom In.
<b>CTRL -</b>	Zoom Out.
<b>F1</b>	Fit to size.
<b>F5</b>	Show or hide the components tree view.

## B Settings

The following describes the available settings of the simulator.

### Settings

The global settings of the simulator specify, among other things, the language, the symbol form to be used or the paths of external tools.

#### Attributes

Use IEEE 91-1984 shapes

Use IEEE 91-1984 shapes instead of rectangular shapes

Language

Language of the GUI. Will only take effect after a restart.

Format

Screen format of expressions.

Color scheme

Color scheme

User Defined Colors

User Defined Colors

Component tree view is visible at startup.

If set, the component tree view is enabled at startup.

Show Grid

Shows a grid in the main window.

Show the number of wires on a bus.

CAUTION: The value is only updated when the simulation starts.

No tool tips for components on the main panel.

If set, no tool tips for the components on the main panel are displayed. Especially in a presentation, these tool tips can be very annoying.

Wire tool tips

If set, lines are highlighted when the mouse hovers over them.

Library

Folder which contains the library with predefined sub circuits. Contains, for example, the components of the 74xx series. You also can add your own circuits by storing them at this location. It must be ensured that the names of all files in this folder and all subfolders are unique.

Java library

A jar file containing additional components implemented in Java.

ATF15xx Fitter

Path to the fitter for the ATF15xx. Enter the directory which contains the fit15xx.exe files provided by Microchip (former ATMEL).

ATMISP

Path to the executable file ATMISP.exe. If set, the ATMISP software can be started automatically!

GHDL

Path to the executable ghdl file. Only necessary if you want to use ghdl to simulate components defined with VHDL.

**IVerilog**

Path to the Icarus Verilog installation folder. Only necessary if you want to use iverilog to simulate components defined with Verilog.

**Toolchain Configuration**

Used to configure an integration of a toolchain. Allows the start of external tools, e.g. to program an FPGA or similar.

**Menus Font Size [%]**

Size of the fonts used in the menu in percent of the default size.

**Use macOS mouse clicks.**

Uses CTRL-click instead of right-click.

**Use Equals-Key**

Use the equal key instead of the plus key. This is always useful if the plus character is not a primary key, but the second assignment of the equals character, e.g. for an American or French keyboard layout.

**Show dialog for automatic renaming of tunnels.**

If set, a dialog for automatically renaming all tunnels of the same name is displayed after a tunnel has been renamed.

## Circuit specific settings

The circuit specific settings affect the behavior of the currently open circuit. For example, the shape that represents the circuit when it is embedded in other circuits. These settings are stored together with the circuit.

**Attributes****Label**

The name of this element.

**Width**

Width of symbol if this circuit is used as a component in another circuit.

**Background color**

Background color of the circuit when it is embedded in another circuit. Is not used for DIL packages.

**Description**

A short description of this element and its usage.

**Modification locked**

The circuit is locked. It is possible to configure diodes and FGF-FETs.

**Shape**

The shape to be used for the representation of the circuit in an embedding circuit. In the "Simple" mode, the inputs are displayed on the left and the outputs on the right side of a simple rectangle. With "Layout", the position of the inputs and outputs and their orientation in the circuit determines the position of the pins. Here it is possible to have pins at the top or the bottom. When selecting "DIL-Chip", a DIL housing is used to display the circuit. The pin numbers of the inputs and outputs determine the position of the pins in this case.

**Custom Shape**

Import of a SVG file

**Height**

Height of symbol if this circuit is used as a component in another circuit.

**Number of DIL pins**

Number of pins. A zero means that the number of pins is determined automatically.

### Content of ROMs

- Content of all used ROMs

### Show measurement values at simulation start

- When the simulation is started, a table with the measured values is shown.

### Show measurement graph at simulation start

- When the simulation is started, a graph with the measured values is shown.

### Show measurement graph in single gate step mode at simulation start

- When the simulation is started, a graph with the measured values in the gate step mode is shown. All gate changes are included in the graph.

### Max number of steps to show

- The maximal number of values stored. If the maximum number is reached, the oldest values are discarded.

### Preload program memory at startup.

- When simulating a processor that uses a RAM device as the program memory, it is difficult to start this processor because the RAM contents are always initialized with zeros at the start of the simulation. This setting allows loading data into the program memory at startup. The program memory in the simulation must be marked as such.

### Program file

- File which should be loaded into the program memory at the start of the simulation.

### Skip in Verilog/VHDL export

- Skips generating the internals of the circuit in Verilog/VHDL export. The references to the circuit are kept, making it possible to override the implementation.

### Circuit is generic

- Allows to create a generic circuit.

## C Command Line Interface

```
java -cp Digital.jar CLI
```

```
test -circ [String] [-tests [String]] [-allowMissingInputs] [-verbose]:
```

The first file name specifies the circuit to be tested. If a second file name is specified, the test cases are executed from this file. If no second file name is specified, the tests are executed from the first file.

Options:

```
-circ [String(def: )]
```

Name of the file to be tested.

```
[-tests [String(def: )]]
```

Name of a file with test cases.

```
[-allowMissingInputs(def: false)]
```

Allows the lack of inputs in the circuit which are defined in the test case. This can be useful if there are several possible solutions which may depend on different inputs.

```
[-verbose(def: false)]
```

If set, the value table is output in case of an error.

```
svg -dig [String] [-svg [String]] [-ieee] [-LaTeX] [-pinsInMathMode] [-hideTest] [-noShapeFilling] [-smallIO] [-noPinMarker] [-thinnerLines] [-highContrast] [-monochrome]:
```

Can be used to create an SVG file from a circuit.

Options:

```
-dig [String(def: )]
```

The file name of the circuit.

```
[-svg [String(def: )]]
```

The name of the SVG file to be written.

```
[-ieee(def: false)]
```

Use the IEEE symbols.

```
[-LaTeX(def: false)]
```

Text is inserted in LaTeX notation. Inkscape is required for further processing.

```
[-pinsInMathMode(def: false)]
```

For pin labels, use math mode even if no indexes are contained.

```
[-hideTest(def: false)]
```

Hide Test Cases

```
[-noShapeFilling(def: false)]
```

Polygons are not filled.

```
[-smallIO(def: false)]
```

Inputs and outputs are represented as small circles.

`[-noPinMarker(def: false)]`

The blue and red pin markers on the symbols are omitted.

`[-thinnerLines(def: false)]`

If set, the lines are drawn slightly thinner.

`[-highContrast(def: false)]`

The wires and the text of the pins are displayed in black.

`[-monochrome(def: false)]`

Only gray colors are used.

`stats -dig [String] [-csv [String]]:`

Creates a CSV file which contains the circuit statistics. All components used are listed in the CSV file.

Options:

`-dig [String(def: )]`

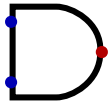
File name of the circuit.

`[-csv [String(def: )]]`

Name of the CSV file to be created. If this option is missing, the table is written to stdout.

## D Components

### 1. Logic



#### 1.1. And

Binary AND gate. Returns high only if all inputs are also set high. It is also possible to use buses with several bits as inputs and output. In this case, a bitwise AND is executed. This means that the lowest bits of all inputs are connected with AND and is output as the lowest bit at the output. The same happens with bit 1, bit 2 and so on. Exportable to VHDL/Verilog.

Inputs

In\_1

The 1. input value for the logic operation.

In\_2

The 2. input value for the logic operation.

Outputs

out

Returns the result of the logic operation.

Attributes

Data Bits

Number of data bits used.

Number of Inputs

The Number of Inputs used. Every input needs to be connected.

Inverted inputs

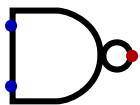
You can select the inputs that are to be inverted.

Rotation

The orientation of the Element in the circuit.

Wide Shape

Uses a wider shape to visualize the gate.



#### 1.2. NAnd

A combination of AND and NOT. Returns 0 only if all inputs are set to 1. If one of the inputs is set to 0 the output is set to 1. It is also possible to use buses with several bits per input. In this case, the operation is applied to each bit of the inputs. Exportable to VHDL/Verilog.

### Inputs

In\_1

The 1. input value for the logic operation.

In\_2

The 2. input value for the logic operation.

### Outputs

out

Returns the result of the logic operation.

### Attributes

Data Bits

Number of data bits used.

Number of Inputs

The Number of Inputs used. Every input needs to be connected.

Inverted inputs

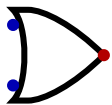
You can select the inputs that are to be inverted.

Rotation

The orientation of the Element in the circuit.

Wide Shape

Uses a wider shape to visualize the gate.



## 1.3. Or

Binary OR gate. Returns a 1 if one of the inputs is set to 1. If all inputs are set to 0 the output is also set to 0. It is also possible to use buses with several bits as inputs and output. In this case, a bitwise OR is executed. This means that the lowest bits of all inputs are connected with OR and is output as the lowest bit at the output. The same happens with bit 1, bit 2 and so on. Exportable to VHDL/Verilog.

### Inputs

In\_1

The 1. input value for the logic operation.

In\_2

The 2. input value for the logic operation.

### Outputs

out

Returns the result of the logic operation.

### Attributes

Data Bits

Number of data bits used.

Number of Inputs

The Number of Inputs used. Every input needs to be connected.



**Inverted inputs**

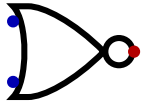
You can select the inputs that are to be inverted.

**Rotation**

The orientation of the Element in the circuit.

**Wide Shape**

Uses a wider shape to visualize the gate.

**1.4. NOR**

A combination of OR and NOT. Returns a 0 if one of the inputs is set to 1. If all inputs are set to 0 the output is also set to 1. It is also possible to use buses with several bits per input. In this case, the operation is applied to each bit of the inputs. Exportable to VHDL/Verilog.

**Inputs**

In\_1

The 1. input value for the logic operation.

In\_2

The 2. input value for the logic operation.

**Outputs**

out

Returns the result of the logic operation.

**Attributes****Data Bits**

Number of data bits used.

**Number of Inputs**

The Number of Inputs used. Every input needs to be connected.

**Inverted inputs**

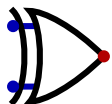
You can select the inputs that are to be inverted.

**Rotation**

The orientation of the Element in the circuit.

**Wide Shape**

Uses a wider shape to visualize the gate.

**1.5. XOR**

If two inputs are used, the output is 0 if both input bits are equal. Otherwise the output is set to 1. If more than two inputs are used, it behaves like cascaded XOR gates (  $A \text{ XOR } B \text{ XOR } C = (A \text{ XOR } B) \text{ XOR } C$  ). It is also possible to use buses with several bits per input. In this case, the operation is applied to each bit of the inputs. Exportable to VHDL/Verilog.

### Inputs

In\_1

The 1. input value for the logic operation.

In\_2

The 2. input value for the logic operation.

### Outputs

out

Returns the result of the logic operation.

### Attributes

Data Bits

Number of data bits used.

Number of Inputs

The Number of Inputs used. Every input needs to be connected.

Inverted inputs

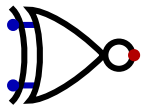
You can select the inputs that are to be inverted.

Rotation

The orientation of the Element in the circuit.

Wide Shape

Uses a wider shape to visualize the gate.



## 1.6. XNOr

A combination of XOR and NOT. The inputs are combined with the XOR operation. The result of this operation is then inverted. It is also possible to use buses with several bits per input. In this case, the operation is applied to each bit of the inputs. Exportable to VHDL/Verilog.

### Inputs

In\_1

The 1. input value for the logic operation.

In\_2

The 2. input value for the logic operation.

### Outputs

out

Returns the result of the logic operation.

### Attributes

Data Bits

Number of data bits used.

Number of Inputs

The Number of Inputs used. Every input needs to be connected.

Inverted inputs

You can select the inputs that are to be inverted.

**Rotation**

The orientation of the Element in the circuit.

**Wide Shape**

Uses a wider shape to visualize the gate.

**1.7. Not**

Inverts the input value. A 1 becomes a 0 and a 0 becomes 1. It is also possible to use a bus with several bits per input. In this case, the operation is applied to each bit of the inputs.  
Exportable to VHDL/Verilog.

**Inputs**

in

The input of the NOT gate.

**Outputs**

out

The inverted input value.

**Attributes****Data Bits**

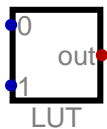
Number of data bits used.

**Rotation**

The orientation of the Element in the circuit.

**Wide Shape**

Uses a wider shape to visualize the gate.

**1.8. Lookup Table**

Gets the output value from a stored table. So this gate can emulate every combinatorial gate.  
Exportable to VHDL/Verilog.

**Inputs**

0

Input 0. This input in combination with all other inputs defines the address of the stored value to be returned.

1

Input 1. This input in combination with all other inputs defines the address of the stored value to be returned.

**Outputs**

out

Returns the stored value at the address set via the inputs.

### Attributes

#### Data Bits

Number of data bits used.

#### Number of Inputs

The Number of Inputs used. Every input needs to be connected.

#### Label

The name of this element.

#### Data

The values stored in this element.

#### Rotation

The orientation of the Element in the circuit.

## 2. IO



### 2.1. Output

Can be used to display an output signal in a circuit. This element is also used to connect a circuit to an embedding circuit. In this case the connection is bidirectional. Is also used to assign a pin number, if the code for a CPLD or FPGA is generated. Exportable to VHDL/Verilog.

#### Inputs

in

This value is used for the output connection.

#### Attributes

##### Data Bits

Number of data bits used.

##### Label

The name of this element.

##### Description

A short description of this element and its usage.

##### Rotation

The orientation of the Element in the circuit.

##### Number Format

The format used to show the numbers.

##### fixed point digits

Number of fractional binary digits

##### Pin number

Number of this pin. Used for the representation of a circuit as a DIL package and the pin assignment when programming a CPLD. If there are several bits, all pin numbers can be specified as a comma-separated list.

##### Show in Measurement Graph

Shows the value in the measurement graph.

##### Small Shape

If selected, a smaller shape will be used.



## 2.2. LED

A LED can be used to visualize an output value. Accepts a single bit. Lights up if the input is set to 1.

### Inputs

**in**  
LED Input. LED lights up if the input is set to 1.

### Attributes

**Label**  
The name of this element.

**Color**  
The Color of the element.

**Rotation**  
The orientation of the Element in the circuit.

**Size**  
The size of the shape in the circuit.



## 2.3. Input

Can be used to interactively manipulate an input signal in a circuit with the mouse. This element is also used to connect a circuit to an embedding circuit. In this case the connection is bidirectional. Is also used to assign an pin number, if code for a CPLD or FPGA is generated. Exportable to VHDL/Verilog.

### Outputs

**out**  
Gives the value which is connected to this input.

### Attributes

**Data Bits**  
Number of data bits used.

**Label**  
The name of this element.

**Description**  
A short description of this element and its usage.

**Rotation**  
The orientation of the Element in the circuit.

**Default**  
This value is set if the circuit is started. A "Z" means high-z state.

**Is three-state input**  
If set the input is allowed to be in high-z state. At the input component this is also allowed if high-z ("Z") is set as the default value.

No zero output.

Avoids zero output. This is especially helpful when setting up relay circuits. Can only be activated if a high-z output is allowed.

Number Format

The format used to show the numbers.

fixed point digits

Number of fractional binary digits

Pin number

Number of this pin. Used for the representation of a circuit as a DIL package and the pin assignment when programming a CPLD. If there are several bits, all pin numbers can be specified as a comma-separated list.

Show in Measurement Graph

Shows the value in the measurement graph.

Small Shape

If selected, a smaller shape will be used.



## 2.4. Clock Input

A clock signal. It's possible to control it by a real-time clock. Depending on the complexity of the circuit, the clock frequency achieved may be less than the selected value. If the frequency is greater than 50Hz, the graphic representation of the circuit will no longer be updated at every clock cycle so that the wire colors will no longer be updated. If the real-time clock is not activated, the clock can be controlled by mouse clicks. Is also used to assign an pin number, if code for a CPLD or FPGA is generated. Exportable to VHDL/Verilog.

Outputs

C

Switches between 0 and 1 with the selected clock frequency.

Attributes

Label

The name of this element.

Start real time clock

If enabled the runtime clock is started when the circuit is started

Frequency/Hz

The real time frequency used for the real time clock

Rotation

The orientation of the Element in the circuit.

Pin number

Number of this pin. Used for the representation of a circuit as a DIL package and the pin assignment when programming a CPLD. If there are several bits, all pin numbers can be specified as a comma-separated list.

Small Shape

If selected, a smaller shape will be used.



## 2.5. Button

A simple push button which goes back to its original state when it is released.

### Outputs

out  
The output signal of the button.

### Attributes

Label  
The name of this element.

Active Low  
If selected the output is low if the component is active.

Map to keyboard  
Button is mapped to the keyboard. To use the cursor keys use UP, DOWN, LEFT or RIGHT as label.

Rotation  
The orientation of the Element in the circuit.

Show in Measurement Graph  
Shows the value in the measurement graph.



## 2.6. DIP Switch

Simple DIP switch that can output either high or low.

### Outputs

out  
The output value of the switch.

### Attributes

Label  
The name of this element.

Description  
A short description of this element and its usage.

Rotation  
The orientation of the Element in the circuit.

Output is High  
The default output value of the DIP switch when the simulation starts.



## 2.7. Probe

A measurement value which can be shown in the data graph or measurement table. This component can be used to easily observe values from embedded circuits. Does not affect the simulation.

Inputs

in  
The measurement value.

Attributes

Label  
The name of this element.

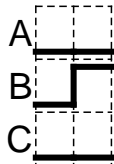
Display Mode  
Defines whether the value or a counter is to be displayed.

Rotation  
The orientation of the Element in the circuit.

Number Format  
The format used to show the numbers.

fixed point digits  
Number of fractional binary digits

Show in Measurement Graph  
Shows the value in the measurement graph.



## 2.8. Data Graph

Shows a data plot inside of the circuit panel. You can plot complete clock cycles or single gate changes. Does not affect the simulation.

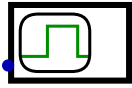
Attributes

Show single gate steps  
Shows all single step steps in the graphic.

Max number of steps to show  
The maximal number of values stored. If the maximum number is reached, the oldest values are discarded.

Snap To Grid  
If set, the component is aligned with the grid.





## 2.9. Triggered Data Graph

Shows a graph of measured values, whereby measured values are only stored if the input signal changes. Storing takes place when the circuit has stabilized. The trigger does not start the measurement like in a real scope, but each trigger event stores a single measurement value for each of the shown signals. As direct input there is only the trigger. The inputs and outputs of the circuit, flip-flops and registers and the probe component can be used as signals. This can be activated in the respective components.

Inputs

T

A change at this input causes measured values to be stored.

Attributes

Label

The name of this element.

Trigger

Trigger condition for data recording.

Max number of steps to show

The maximal number of values stored. If the maximum number is reached, the oldest values are discarded.

## 3. IO - Displays



### 3.1. RGB-LED

An RGB LED whose color can be controlled via three inputs. At each of the three inputs, a color channel is connected.

Inputs

R

The red color channel.

G

The green color channel.

B

The blue color channel.

Attributes

Data Bits

Number of data bits used.

Label

The name of this element.

**Rotation**

The orientation of the Element in the circuit.

**Size**

The size of the shape in the circuit.

**3.2. LED with two connections.**

LED with connections for the cathode and the anode. The LED lights up if the anode is connected to high and the cathode is connected to low. This LED cannot be used as a pull-down resistor. It acts solely as a display element. The shown resistor is only meant to symbolize the required series resistor to limit the current.

**Inputs**

A

The anode connection of the LED.

C

The cathode connection of the LED.

**Attributes****Label**

The name of this element.

**Color**

The Color of the element.

**Rotation**

The orientation of the Element in the circuit.

**3.3. Button with LED**

A simple push button which goes back to its original state when it is released. The push button has an LED which can be switched via an input signal.

**Inputs**

in

Input for controlling the LED.

**Outputs**

out

The output signal of the button.

**Attributes****Label**

The name of this element.

**Active Low**

If selected the output is low if the component is active.

**Map to keyboard**

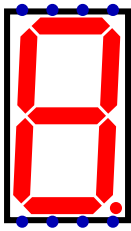
Button is mapped to the keyboard. To use the cursor keys use UP, DOWN, LEFT or RIGHT as label.

**Color**

The Color of the element.

**Rotation**

The orientation of the Element in the circuit.



### 3.4. Seven-Segment Display

Seven Segment Display, every segment has its own control input.

**Inputs**

- a This input controls the upper, horizontal line.
- b This input controls the upper, right, vertical line.
- c This input controls the lower, right, vertical line.
- d This input controls the lower horizontal line.
- e This input controls the lower, left, vertical line.
- f This input controls the upper, left, vertical line.
- g This input controls the middle, horizontal line.
- dp This input controls the decimal point.

**Attributes****Color**

The Color of the element.

**Common Connection**

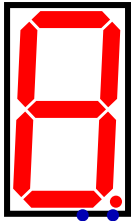
If selected, a common cathode or anode input is also simulated.

**Common**

Kind of common connection.

### Avoid Flicker

It is not possible to increase the frequency so much that the flickering disappears. In order to suppress the flickering nevertheless, a "afterglow" can be switched on for the LEDs with this option. If enabled, the LEDs remain on, even if one of the pins changes to high-z. This simulates a frequency above the critical flicker fusion frequency.



## 3.5. Seven-Segment Hex Display

Seven Segment Display with a 4 bit hex input

### Inputs

- d  
The value at this input is visualized at the display.
- dp  
This input controls the decimal point.

### Attributes

- Color  
The Color of the element.
- Size  
The size of the shape in the circuit.



## 3.6. 16-Segment Display

The LED input has 16 bits which control the segments. The second input controls the decimal point.

### Inputs

- led  
16-bit bus for driving the LEDs.
- dp  
This input controls the decimal point.

### Attributes

Color

The Color of the element.

Size

The size of the shape in the circuit.



### 3.7. Light Bulb

Light bulb with two connections. If a current flows, the bulb lights up! The direction of the current does not matter. The lamp lights when the inputs have different values. The bulb behaves similar to an XOR gate.

Inputs

A

Connection

B

Connection

Attributes

Label

The name of this element.

Color

The Color of the element.

Rotation

The orientation of the Element in the circuit.



### 3.8. LED-Matrix

A matrix of LEDs. The LEDs are shown in a separate window. The LEDs of a column of the display are controlled by a data word. At another input, the current column is selected. So a multiplexed display is realized. The LEDs are able to light up indefinitely in the simulation to prevent the display from flickering.

Inputs

r-data

The row state of the LEDs of a column. Each bit in this data word represents the state of a row of the current column.

c-addr

The number of the current column whose state is currently visible at the other input.

Attributes

Label

The name of this element.

**Rows**

Specifies the number of rows by specifying the number of bits of the row word.

**Address bits of columns**

Addresses the individual columns. Three bits means eight columns.

**Color**

The Color of the element.

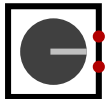
**Avoid Flicker**

It is not possible to increase the frequency so much that the flickering disappears. In order to suppress the flickering nevertheless, a "afterglow" can be switched on for the LEDs with this option. If enabled, the LEDs remain on, even if one of the pins changes to high-z. This simulates a frequency above the critical flicker fusion frequency.

**Rotation**

The orientation of the Element in the circuit.

## 4. IO - Mechanical



### 4.1. Rotary Encoder

Rotary knob with rotary encoder. Used to detect rotational movements.

**Outputs**

A

encoder signal A

B

encoder signal B

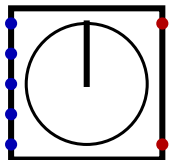
**Attributes**

Label

The name of this element.

Rotation

The orientation of the Element in the circuit.



### 4.2. Stepper Motor, unipolar

Unipolar stepper motor with two limit position switches. Full step drive, half step drive and wave drive are supported.

## Inputs

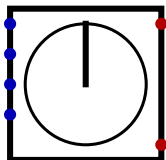
- P0  
Phase 0
- P1  
Phase 1
- P2  
Phase 2
- P3  
Phase 3
- com  
Common center coil connection

## Outputs

- S0  
Limit position switch 0, becomes 1 when the motor angle is 0°.
- S1  
Limit position switch 1, becomes 1 when the motor angle is 180°.

## Attributes

- Label  
The name of this element.
- Inverted output  
If selected the output is inverted.
- Rotation  
The orientation of the Element in the circuit.



### 4.3. Stepper Motor, bipolar

Bipolar stepper motor with two limit position switches. Full step drive, half step drive and wave drive are supported.

## Inputs

- A+  
Coil A, positive
- A-  
Coil A, negative
- B+  
Coil B, positive
- B-  
Coil B, negative

## Outputs

S0

Limit position switch 0, becomes 1 when the motor angle is 0°.

S1

Limit position switch 1, becomes 1 when the motor angle is 180°.

## Attributes

Label

The name of this element.

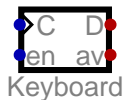
Inverted output

If selected the output is inverted.

Rotation

The orientation of the Element in the circuit.

# 5. IO - Peripherals



## 5.1. Keyboard

A keyboard that can be used to enter text. This component buffers the input, which can then be read out. A separate window is opened for the text input.

### Inputs

C

Clock. A rising edge removes the oldest character from the buffer.

en

If high, the output D is active and one character is output. It also enables the clock input.

### Outputs

D

The last typed character, or zero if no character is available. Output is the 16 bit Java char value.

av

This output indicates that characters are available. It can be used to trigger an interrupt.

### Attributes

Label

The name of this element.

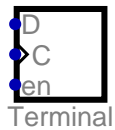
Inverted inputs

You can select the inputs that are to be inverted.

Rotation

The orientation of the Element in the circuit.





## 5.2. Terminal

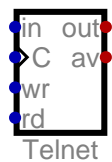
You can write ASCII characters to this terminal. The terminal opens its own window to visualize the output.

### Inputs

- D  
The data to write to the terminal
- C  
Clock. A rising edge writes the value at the input to the terminal window.
- en  
A high at this input enables the clock input.

### Attributes

- Characters per line  
The number of characters shown in a single line.
- Lines  
The number of lines to show.
- Label  
The name of this element.
- Rotation  
The orientation of the Element in the circuit.



## 5.3. Telnet

Allows a Telnet connection to the circuit. It is possible to receive and send characters via Telnet.

### Inputs

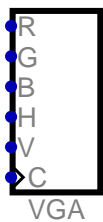
- in  
The data to be sent.
- C  
Clock input
- wr  
If set, the input data byte is sent.
- rd  
If set, a received byte is output.

## Outputs

- out  
Data output
- av  
Outputs a one if data is present.

## Attributes

- Label  
The name of this element.
- Telnet mode  
If set, the Telnet control commands are evaluated. In addition, the server sends the SGA and ECHO commands. If this option is disabled, the server is a simple TCP server.
- Port  
The port to be opened by the server.
- Rotation  
The orientation of the Element in the circuit.



## 5.4. VGA Monitor

Analyzes the incoming video signals and displays the corresponding graphic. Since the simulation cannot run in real time, the pixel clock is required in addition to the video signals.

### Inputs

- R  
The red color component
- G  
The green color component
- B  
The blue color component
- H  
The horizontal synchronization signal
- V  
The vertical synchronization signal
- C  
The pixel clock

### Attributes

- Label  
The name of this element.
- Rotation  
The orientation of the Element in the circuit.



## 5.5. MIDI

Uses the MIDI system to play notes.

Inputs

- N  
Note
- V  
Volume
- OnOff  
If set, this translates to pressing a keyboard key (key down event), if not set, this translates to releasing the key (key up event).
- en  
Enables the component
- C  
Clock

Attributes

- Label  
The name of this element.
- MIDI channel  
Selects the MIDI channel to use.
- MIDI instrument  
The MIDI instrument to use.
- Allow program change  
Adds a new input PC. If this input is set to high, the value at input N is used to change the program (instrument).
- Rotation  
The orientation of the Element in the circuit.

## 6. Wires



### 6.1. Ground

A connection to ground. Output is always zero. Exportable to VHDL/Verilog.

Outputs

- out  
Output always returns 0.

#### Attributes

##### Data Bits

Number of data bits used.

##### Label

The name of this element.

##### Rotation

The orientation of the Element in the circuit.



## 6.2. Supply voltage

A connection to the supply voltage. Output is always one. Exportable to VHDL/Verilog.

#### Outputs

##### out

This output always returns 1.

#### Attributes

##### Data Bits

Number of data bits used.

##### Label

The name of this element.

##### Rotation

The orientation of the Element in the circuit.

1•

## 6.3. Constant value

A component which returns a given value as a simple constant value. The value can be set in the attribute dialog. Exportable to VHDL/Verilog.

#### Outputs

##### out

Returns the given value as a constant.

#### Attributes

##### Data Bits

Number of data bits used.

##### Value

The value of the constant.

##### Rotation

The orientation of the Element in the circuit.

##### Number Format

The format used to show the numbers.

##### fixed point digits

Number of fractional binary digits



## 6.4. Tunnel

Connects components without a wire. All tunnel elements, which have the same net name, are connected together. Works only locally, so it is not possible to connect different circuits. Unnamed tunnels are ignored silently. Exportable to VHDL/Verilog.

### Inputs

in

The connection to the tunnel.

### Attributes

Net name

All nets with identical name are connected together.

Rotation

The orientation of the Element in the circuit.



## 6.5. Splitter/Merger

Splits or creates a wire bundle or a data bus with more than one bit. With a bus it is e.g. possible to generate 16-bit connections without having to route 16 individual wires. All 16 connections can be merged into one wire. The splitter has a direction, meaning it can only transmit signals in one direction. Exportable to VHDL/Verilog.

### Inputs

0-3

The input bits 0-3.

4-7

The input bits 4-7.

### Outputs

0-7

The output bits 0-7.

### Attributes

#### Input Splitting

If e.g. four bits, two bits and two further bits are to be used as inputs, this can be configured with "4,2,2". The number indicates the number of bits. For convenience, the asterisk can be used: 16 bits can be configured with "[Bits]\*[Number]" as "1\*16". It is also possible to specify the bits to be used directly and in any order. For example, "4-7,0-3" configures bits 4-7 and 0-3. This notation allows any bit arrangement. The input bits must be specified completely and unambiguously.

**Output splitting**

If e.g. four bits, two bits and two further bits are to be used as outputs, this can be configured with "4,2,2". The number indicates the number of bits. For convenience, the asterisk can be used: 16 bits can be configured with "[Bits]\*[Number]" as "1\*16". It is also possible to specify the bits to be used directly and in any order. For example, "4-7,0-3" configures bits 4-7 and 0-3. This notation allows any bit arrangement. Output bits can also be output several times: "0-7,1-6,4-7".

**Rotation**

The orientation of the Element in the circuit.

**Mirror**

Mirrors the component in the circuit.

**Spreading**

Configures the spread of the inputs and outputs in the circuit.

**6.6. Driver**

A driver can be used to connect a signal value to another wire. The driver is controlled by the sel input. If the sel input is low, the output is in high z state. If the sel input is high, the output is set to the input value. Exportable to VHDL/Verilog.

**Inputs**

in

The input value of the driver.

sel

Pin to control the driver. If its value is 1 the input is set to the output. If the value is 0, the output is in high z state.

**Outputs**

out

If the sel input is 1 the input is given to this output. If the sel input is 0, this output is in high z state.

**Attributes**

Data Bits

Number of data bits used.

Flip selector position

This option allows you to move the selector pin to the opposite side of the plexer.

Rotation

The orientation of the Element in the circuit.



### 6.7. Driver, inverted select

A driver can be used to connect a data word to another line. The driver is controlled by the sel input. If the sel input is high, the output is in high z state. If the sel input is low, the output is set to the input value. Exportable to VHDL/Verilog.

#### Inputs

- in  
The input value of the driver.
- sel  
Pin to control the driver. If its value is 0 the input is given to the output. If the value is 1, the output is in high z state.

#### Outputs

- out  
If the sel input is 1 the input is given to this output. If the sel input is 0, this output is in high z state.

#### Attributes

- Data Bits  
Number of data bits used.
- Flip selector position  
This option allows you to move the selector pin to the opposite side of the plexer.
- Rotation  
The orientation of the Element in the circuit.



### 6.8. Delay

Delays the signal by one propagation delay time. Delays a signal for an adjustable number of gate delays. All other components in Digital have a gate delay of one propagation delay time. This component can be used to realize any necessary propagation delay.

#### Inputs

- in  
Input of the signal to be delayed.

#### Outputs

- out  
The input signal delayed by one gate delay time.

#### Attributes

- Data Bits  
Number of data bits used.

Duration

Delay time in units of the common gate propagation delay.

Rotation

The orientation of the Element in the circuit.



### 6.9. Pull-Up Resistor

If a net is in a HighZ state, this resistor pulls the net to high. In any other case this component has no effect.

Outputs

out

A "weak high".

Attributes

Data Bits

Number of data bits used.

Rotation

The orientation of the Element in the circuit.



### 6.10. Pull-Down Resistor

If the net is in a HighZ state, this resistor pulls the net to ground. In any other case this component has no effect.

Outputs

out

A "weak low".

Attributes

Data Bits

Number of data bits used.

Rotation

The orientation of the Element in the circuit.



### 6.11. Not Connected

This component can be used to set a wire to High-Z. If an input of a logical gate is set to high-Z, the read value is undefined. Note that in reality in many cases excessive current consumption and even damage can occur if a digital input is not set to high or low but remains unconnected.



## Outputs

out

This output always outputs High-Z.

## Attributes

Data Bits

Number of data bits used.

# 7. Plexers



## 7.1. Multiplexer

A component which uses the value of the sel pin to decide which input value is set to the output. Exportable to VHDL/Verilog.

### Inputs

sel

This input is used to select the data input which is output.

in\_0

The 0. data input of the multiplexer.

in\_1

The 1. data input of the multiplexer.

### Outputs

out

The value of the selected input.

### Attributes

Data Bits

Number of data bits used.

Number of Selector Bits

Number of bits used for the selector input.

Flip selector position

This option allows you to move the selector pin to the opposite side of the plexer.

Rotation

The orientation of the Element in the circuit.



## 7.2. Demultiplexer

A component that can output the input value to one of the outputs. The other outputs are set to the default value. Exportable to VHDL/Verilog.

### Inputs

sel

This pin selects the output to use.

in

The value of this input is given to the selected data output.

### Outputs

out\_0

Data output 0.

out\_1

Data output 1.

### Attributes

Data Bits

Number of data bits used.

Number of Selector Bits

Number of bits used for the selector input.

Flip selector position

This option allows you to move the selector pin to the opposite side of the plexer.

Rotation

The orientation of the Element in the circuit.

Default

This value is set if the circuit is started. At the demultiplexer, this value is set for the non-selected outputs.



## 7.3. Decoder

One selectable output pin is 1, all other outputs are set to 0. Exportable to VHDL/Verilog.

### Inputs

sel

This input selects the enabled output. The selected output is set to 1. All other outputs are set to 0.

### Outputs

out\_0

Output 0. This output is 1 if selected by the sel input.

out\_1

Output 1. This output is 1 if selected by the sel input.

### Attributes

Number of Selector Bits

Number of bits used for the selector input.

Flip selector position

This option allows you to move the selector pin to the opposite side of the plexer.

**Rotation**

The orientation of the Element in the circuit.

**7.4. Bit Selector**

Selects a single bit from a data bus. Exportable to VHDL/Verilog.

**Inputs**

in

The input bus

sel

This input selects the bit

**Outputs**

out

The selected bit.

**Attributes**

Number of Selector Bits

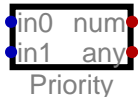
Number of bits used for the selector input.

Flip selector position

This option allows you to move the selector pin to the opposite side of the plexer.

Rotation

The orientation of the Element in the circuit.

**7.5. Priority Encoder**

If one of the inputs is set, its number is output. If several inputs are set at the same time, the highest number is output. Exportable to VHDL/Verilog.

**Inputs**

in0

The 0. input of the priority encoder.

in1

The 1. input of the priority encoder.

**Outputs**

num

Number of the set input.

any

If this output is set, at least one of the inputs is set.

**Attributes**

**Label**

The name of this element.

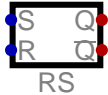
**Number of Selector Bits**

Number of bits used for the selector input.

**Rotation**

The orientation of the Element in the circuit.

## 8. Flip-Flops



### 8.1. RS-Flip-flop

A component to store a single bit. Provides the functions "set" and "reset" to set or reset the stored bit. If both inputs are switched to one, both outputs also output a one. If both inputs switch back to zero at the same time, the final state is random.

**Inputs**

S

The set input.

R

The reset input.

**Outputs**

Q

Returns the stored value.

$\neg Q$

Returns the inverted stored value.

**Attributes****Label**

The name of this element.

**Inverted inputs**

You can select the inputs that are to be inverted.

**Rotation**

The orientation of the Element in the circuit.

**Mirror**

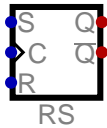
Mirrors the component in the circuit.

**Default**

This value is set if the circuit is started. At the demultiplexer, this value is set for the non-selected outputs.

**Use as measurement value**

If set the value is a measurement value and appears in the graph and data table. In addition, a label must be specified that can serve as identification of the value.



## 8.2. RS-Flip-flop, clocked

A component to store a single bit. Provides the functions "set" and "reset" to set or reset the stored bit. If both inputs (S, R) are set at the rising edge of the clock, the final state is random.

### Inputs

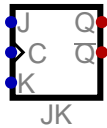
- S  
The set input.
- C  
The clock input. A rising edge initiates a state transition.
- R  
The reset input.

### Outputs

- Q  
Returns the stored value.
- $\neg Q$   
Returns the inverted stored value.

### Attributes

- Label  
The name of this element.
- Inverted inputs  
You can select the inputs that are to be inverted.
- Rotation  
The orientation of the Element in the circuit.
- Mirror  
Mirrors the component in the circuit.
- Default  
This value is set if the circuit is started. At the demultiplexer, this value is set for the non-selected outputs.
- Use as measurement value  
If set the value is a measurement value and appears in the graph and data table. In addition, a label must be specified that can serve as identification of the value.



### 8.3. JK-Flip-flop

Has the possibility to store ( $J=K=0$ ), set ( $J=1, K=0$ ), reset ( $J=0, K=1$ ) or toggle ( $J=K=1$ ) the stored value. A change of state takes place only at a rising edge at the clock input C. Exportable to VHDL/Verilog.

#### Inputs

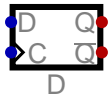
- J  
The set input of the flip-flop.
- C  
The clock input. A rising edge initiates a state change.
- K  
The reset input of the flip-flop.

#### Outputs

- Q  
Returns the stored value.
- $\neg Q$   
Returns the inverted stored value.

#### Attributes

- Label  
The name of this element.
- Inverted inputs  
You can select the inputs that are to be inverted.
- Rotation  
The orientation of the Element in the circuit.
- Mirror  
Mirrors the component in the circuit.
- Default  
This value is set if the circuit is started. At the demultiplexer, this value is set for the non-selected outputs.
- Use as measurement value  
If set the value is a measurement value and appears in the graph and data table. In addition, a label must be specified that can serve as identification of the value.



### 8.4. D-Flip-flop

A component used to store a value. The value on pin D is stored on a rising edge of the clock pin C. The bit width can be selected, which allows to store multiple bits. Exportable to VHDL/Verilog.

#### Inputs

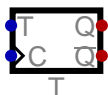
- D  
Input of the bit to be stored.
- C  
Clock pin to store a value. The value on input D is stored on a rising edge of this pin.

#### Outputs

- Q  
Returns the stored value.
- $\neg Q$   
Returns the inverted stored value.

#### Attributes

- Data Bits  
Number of data bits used.
- Label  
The name of this element.
- Inverted inputs  
You can select the inputs that are to be inverted.
- Rotation  
The orientation of the Element in the circuit.
- Mirror  
Mirrors the component in the circuit.
- Default  
This value is set if the circuit is started. At the demultiplexer, this value is set for the non-selected outputs.
- Use as measurement value  
If set the value is a measurement value and appears in the graph and data table. In addition, a label must be specified that can serve as identification of the value.



### 8.5. T-Flip-Flop

Stores a single bit. Toggles the state on a rising edge at input C.

## Inputs

T

Enables the toggle function.

C

Clock input. A rising edge toggles the output, if input T is set to 1.

## Outputs

Q

Returns the stored value.

 $\neg Q$ 

Returns the inverted stored value.

## Attributes

Label

The name of this element.

Enable Input

If set an enable input (T) is available.

Inverted inputs

You can select the inputs that are to be inverted.

Rotation

The orientation of the Element in the circuit.

Mirror

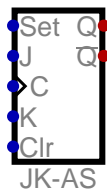
Mirrors the component in the circuit.

Default

This value is set if the circuit is started. At the demultiplexer, this value is set for the non-selected outputs.

Use as measurement value

If set the value is a measurement value and appears in the graph and data table. In addition, a label must be specified that can serve as identification of the value.



## 8.6. JK-Flip-flop, asynchronous

Has the possibility to store ( $J=K=0$ ), set ( $J=1, K=0$ ), reset ( $J=0, K=1$ ) or toggle ( $J=K=1$ ) the stored value. A change of state takes place only at a rising edge at the clock input C. There are two additional inputs which set or reset the state immediately without a clock signal. Exportable to VHDL/Verilog.



## Inputs

## Set

asynchronous set. A high value at this input sets the flip-flop.

## J

The set input of the flip-flop.

## C

The Clock input. A rising edge initiates a state change.

## K

The reset input of the flip-flop.

## Clr

asynchronous clear. A high value at this input clears the flip-flop.

## Outputs

## Q

Returns the stored value.

 $\neg Q$ 

Returns the inverted stored value.

## Attributes

## Label

The name of this element.

## Inverted inputs

You can select the inputs that are to be inverted.

## Rotation

The orientation of the Element in the circuit.

## Mirror

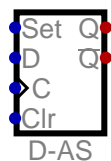
Mirrors the component in the circuit.

## Default

This value is set if the circuit is started. At the demultiplexer, this value is set for the non-selected outputs.

## Use as measurement value

If set the value is a measurement value and appears in the graph and data table. In addition, a label must be specified that can serve as identification of the value.



## 8.7. D-Flip-flop, asynchronous

A component used to store a value. The value on pin D is stored on a rising edge of the clock pin C. There are two additional inputs which set or reset the state immediately without a clock signal. The bit width can be selected, which allows to store multiple bits. Exportable to VHDL/Verilog.

## Inputs

## Set

asynchronous set. If set to one, all stored bits are set to one.

## D

Input of the bit to be stored.

## C

Control pin to store a bit. The bit on input D is stored on a rising edge of this pin.

## Clr

asynchronous clear. If set to one, all stored bits are set to zero.

## Outputs

## Q

Returns the stored value.

 $\neg Q$ 

Returns the inverted stored value.

## Attributes

## Data Bits

Number of data bits used.

## Label

The name of this element.

## Inverted inputs

You can select the inputs that are to be inverted.

## Rotation

The orientation of the Element in the circuit.

## Mirror

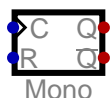
Mirrors the component in the circuit.

## Default

This value is set if the circuit is started. At the demultiplexer, this value is set for the non-selected outputs.

## Use as measurement value

If set the value is a measurement value and appears in the graph and data table. In addition, a label must be specified that can serve as identification of the value.



## 8.8. Monoflop

The monoflop is set at a rising edge at the clock input. After a configurable delay time, the monoflop will be cleared automatically. The monoflop is retriggerable. It can only be used if there is exactly one clock component present in the circuit. This clock component is used as time base to measure the time delay.

## Inputs

C

The Clock input. A rising edge sets the monoflop.

R

Reset Input. A high value clears the monoflop.

## Outputs

Q

output

 $\neg Q$ 

inverted output

## Attributes

Label

The name of this element.

Pulse Width

The pulse width is measured in clock cycles.

Inverted inputs

You can select the inputs that are to be inverted.

Rotation

The orientation of the Element in the circuit.

Mirror

Mirrors the component in the circuit.

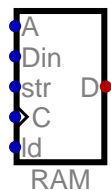
Default

This value is set if the circuit is started. At the demultiplexer, this value is set for the non-selected outputs.

Use as measurement value

If set the value is a measurement value and appears in the graph and data table. In addition, a label must be specified that can serve as identification of the value.

## 9. Memory - RAM



### 9.1. RAM, separated Ports

A RAM module with separate inputs for storing and output for reading the stored data. Exportable to VHDL/Verilog.

### Inputs

A

The address to read from or write to.

Din

The data to be stored in the RAM.

str

If this input is high and when the clock becomes high, the data is stored.

C

Clock input

Id

If this input is high the output is activated and the data is visible at the output.

### Outputs

D

The data output pin

### Attributes

Data Bits

Number of data bits used.

Address Bits

Number of address bits used.

Label

The name of this element.

Rotation

The orientation of the Element in the circuit.

Number Format

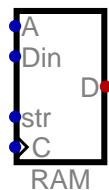
The format used to show the numbers.

fixed point digits

Number of fractional binary digits

Program Memory

Makes this ROM to program memory. So it can be accessed by an external IDE.



## 9.2. Block-RAM, separated ports

A RAM module with separate inputs for storing and output for reading the stored data. This RAM only updates its output on a rising edge of the clock input. This allows the usage of Block RAM on an FPGA. Exportable to VHDL/Verilog.

### Inputs

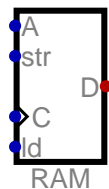
- A  
The address to read from or write to.
- Din  
The data to be stored in the RAM.
- str  
If this input is high and when the clock becomes high, the data is stored.
- C  
Clock input

### Outputs

- D  
The data output pin

### Attributes

- Data Bits  
Number of data bits used.
- Address Bits  
Number of address bits used.
- Label  
The name of this element.
- Rotation  
The orientation of the Element in the circuit.
- Program Memory  
Makes this ROM to program memory. So it can be accessed by an external IDE.



## 9.3. RAM, bidirectional Port

A RAM module with a bidirectional pin for reading and writing the data.

### Inputs

- A  
The address to read and write.
- str  
If this input is high when the clock becomes high, the data is stored.
- C  
Clock
- ld  
If this input is high the output is activated and the data is visible at the output.

### Outputs

- D  
The bidirectional data connection.

## Attributes

## Data Bits

Number of data bits used.

## Address Bits

Number of address bits used.

## Label

The name of this element.

## Rotation

The orientation of the Element in the circuit.

## Number Format

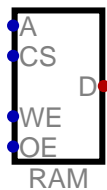
The format used to show the numbers.

## fixed point digits

Number of fractional binary digits

## Program Memory

Makes this ROM to program memory. So it can be accessed by an external IDE.



## 9.4. RAM, Chip Select

A RAM module with a bidirectional connection for reading and writing the data. If the CS input is low, the component is disabled. This allows to build a larger RAM from some smaller RAMs and a address decoder. The write cycle works as follows: By setting CS to high, the component is selected. A rising edge at WE latches the address, and the following falling edge at WE stores the data.

## Inputs

## A

The address to read and write.

## CS

If this input is high, this RAM is enabled. Otherwise the output is always in high Z state.

## WE

If set to high the data is written to the RAM.

## OE

If this input is high, the stored value is output.

## Outputs

## D

The bidirectional data connection.

## Attributes

## Data Bits

Number of data bits used.

**Address Bits**

Number of address bits used.

**Label**

The name of this element.

**Inverted inputs**

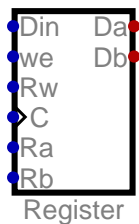
You can select the inputs that are to be inverted.

**Rotation**

The orientation of the Element in the circuit.

**Program Memory**

Makes this ROM to program memory. So it can be accessed by an external IDE.

**9.5. Register File**

Memory with one port that allows to write and two ports that allow to read from the memory simultaneously. Can be used to implement processor registers. Two registers can be read simultaneously and a third can be written. Exportable to VHDL/Verilog.

**Inputs****Din**

The data to be stored in the register Rw.

**we**

If this input is high and when the clock becomes high, the data is stored.

**Rw**

The register into which the data is written.

**C**

Clock

**Ra**

The register which is visible at port a.

**Rb**

The register which is visible at port b.

**Outputs****Da**

Output Port a

**Db**

Output Port b

**Attributes****Data Bits**

Number of data bits used.

**Address Bits**

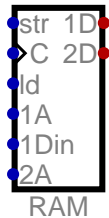
Number of address bits used.

**Label**

The name of this element.

**Rotation**

The orientation of the Element in the circuit.

**9.6. RAM, Dual Port**

RAM with one port that allows to write to and read from the RAM, and a second read only port. This second port can be used to give some graphic logic access to the memory contents. In this way, a processor can write to the RAM, and a graphics logic can simultaneously read from the RAM. Exportable to VHDL/Verilog.

**Inputs**

**str**

If this input is high and when the clock becomes high, the data is stored.

**C**

Clock

**ld**

If this input is high the output is activated and the data is visible at the output 1D.

**1A**

The address at which port 1 is read or written.

**1Din**

The data to be stored in the RAM.

**2A**

The address used to read via port 2.

**Outputs**

**1D**

Output Port 1

**2D**

Output Port 2

**Attributes**

**Data Bits**

Number of data bits used.

**Address Bits**

Number of address bits used.

**Label**

The name of this element.

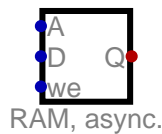
**Rotation**

The orientation of the Element in the circuit.

**Program Memory**

Makes this ROM to program memory. So it can be accessed by an external IDE.





### 9.7. RAM, async.

As long as we is set, it is stored. Corresponds to a very simple RAM, where the address and data lines are directly connected to the decoders of the memory cells. Exportable to VHDL/Verilog.

#### Inputs

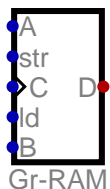
- A  
The address at which reading or writing takes place.
- D  
The data to be stored.
- we  
Write enable. As long as this input is set to 1, the value applied to D is stored at the address applied to A whenever A or D is changed.

#### Outputs

- Q  
Output of the stored data.

#### Attributes

- Data Bits  
Number of data bits used.
- Address Bits  
Number of address bits used.
- Inverted inputs  
You can select the inputs that are to be inverted.
- Label  
The name of this element.
- Rotation  
The orientation of the Element in the circuit.
- Program Memory  
Makes this ROM to program memory. So it can be accessed by an external IDE.



### 9.8. Graphic RAM

Used to show a bitmap graphic. This element behaves like a RAM. In addition it shows its content on a graphic screen. Every pixel is represented by a memory address. The value stored defines the color of the pixel, using a fixed color palette. There are two screen buffers

implemented to support page flipping. The input B selects which buffer is shown. Thus, the total memory size is  $dx * dy * 2$  words. The palette used is structured as follows: The indices 0-9 correspond to the colors white, black, red, green, blue, yellow, cyan, magenta, orange and pink. The indices 32-63 map gray values and the indices 64-127 represent 64 color values each with two bits per color channel. This results in a simple palette that can be addressed with only 7-bit. If the architecture supports a 16-bit index, from Index 0x8000, a high-color mode with 5 bits per color channel can be used, which enables 32768 colors.

#### Inputs

- A  
The address to read and write.
- str  
If this input is high when the clock becomes high, the data is stored.
- C  
Clock
- ld  
If this input is high the output is activated and the data is visible at the output.
- B  
Selects the screen buffer to show.

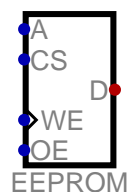
#### Outputs

- D  
The bidirectional data connection.

#### Attributes

- Data Bits  
Number of data bits used.
- Label  
The name of this element.
- Width in pixels  
The screen width in pixels.
- Height in pixels  
The screen height in pixels.
- Rotation  
The orientation of the Element in the circuit.

## 10. Memory - EEPROM



### 10.1. EEPROM

A EEPROM module with a bidirectional connection for reading and writing the data. If the CS input is low, the component is disabled. The data content is stored like in a ROM. It is thus preserved when the simulation is terminated and restarted. The write cycle works as follows:

By setting CS to high, the component is selected. A rising edge at WE latches the address, and the following falling edge at WE stores the data.

#### Inputs

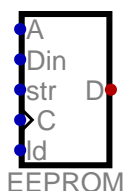
- A  
The address to read and write.
- CS  
If this input is high, this EEPROM is enabled. Otherwise the output is always in high Z state.
- WE  
If set to high the data is written to the EEPROM.
- OE  
If this input is high, the stored value is output.

#### Outputs

- D  
The bidirectional data connection.

#### Attributes

- Data Bits  
Number of data bits used.
- Address Bits  
Number of address bits used.
- Label  
The name of this element.
- Inverted inputs  
You can select the inputs that are to be inverted.
- Data  
The values stored in this element.
- Rotation  
The orientation of the Element in the circuit.
- Number Format  
The format used to show the numbers.
- fixed point digits  
Number of fractional binary digits
- Program Memory  
Makes this ROM to program memory. So it can be accessed by an external IDE.



## 10.2. EEPROM, separated Ports

A EEPROM module with separate inputs for storing and output for reading the stored data.

### Inputs

A

The address to read from or write to.

Din

The data to be stored in the EEPROM.

str

If this input is high and when the clock becomes high, the data is stored.

C

Clock input

ld

If this input is high the output is activated and the data is visible at the output.

### Outputs

D

The data output pin

### Attributes

Data Bits

Number of data bits used.

Address Bits

Number of address bits used.

Label

The name of this element.

Data

The values stored in this element.

Rotation

The orientation of the Element in the circuit.

Number Format

The format used to show the numbers.

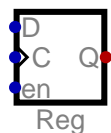
fixed point digits

Number of fractional binary digits

Program Memory

Makes this ROM to program memory. So it can be accessed by an external IDE.

## 11. Memory



### 11.1. Register

A component to store values. The bit width of the data word can be selected. Unlike a D flip-flop, the register provides an input which enables the clock. Exportable to VHDL/Verilog.

### Inputs

D

Input pin of the data word to be stored.

C

Clock input. A rising edge stores the value at the D pin.

en

Enable pin. Storing a value works only if this pin is set high.

### Outputs

Q

Returns the stored value.

### Attributes

Data Bits

Number of data bits used.

Label

The name of this element.

Inverted inputs

You can select the inputs that are to be inverted.

Rotation

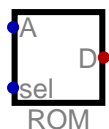
The orientation of the Element in the circuit.

Program Counter

Makes this register a program counter. The value of this register is returned to the external assembler IDE to mark the current line of code during debugging.

Use as measurement value

If set the value is a measurement value and appears in the graph and data table. In addition, a label must be specified that can serve as identification of the value.



## 11.2. ROM

A non-volatile memory component. The stored data can be edited in the attributes dialog. Exportable to VHDL/Verilog.

### Inputs

A

This pin defines the address of data word to be output.

sel

If the input is high, the output is activated. If it is low, the data output is in high Z state.

### Outputs

D

The selected data word if the sel input is high.

### Attributes

**Data Bits**

Number of data bits used.

**Address Bits**

Number of address bits used.

**Label**

The name of this element.

**Data**

The values stored in this element.

**Rotation**

The orientation of the Element in the circuit.

**Number Format**

The format used to show the numbers.

**fixed point digits**

Number of fractional binary digits

**Program Memory**

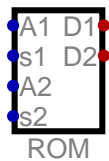
Makes this ROM to program memory. So it can be accessed by an external IDE.

**Reload at model start**

Reloads the HEX file every time the model is started.

**File**

File to be loaded into the ROM.

**11.3. ROM dual port**

A non-volatile memory component. The stored data can be edited in the attributes dialog.

**Inputs****A1**

This pin defines the address of data word to be output on D1.

**s1**

If the input is high, the output D1 is activated. If it is low, the data output is in high Z state.

**A2**

This pin defines the address of data word to be output on D2.

**s2**

If the input is high, the output D2 is activated. If it is low, the data output is in high Z state.

**Outputs****D1**

The selected data word if the s1 input is high.

**D2**

The selected data word if the s2 input is high.

**Attributes**

**Data Bits**

Number of data bits used.

**Address Bits**

Number of address bits used.

**Label**

The name of this element.

**Data**

The values stored in this element.

**Rotation**

The orientation of the Element in the circuit.

**Number Format**

The format used to show the numbers.

**fixed point digits**

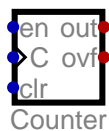
Number of fractional binary digits

**Program Memory**

Makes this ROM to program memory. So it can be accessed by an external IDE.

**Reload at model start**

Reloads the HEX file every time the model is started.

**11.4. Counter**

A simple counter component. The clock input increases the counter. Can be reset back to 0 with the clr input. The number of bits can be set in the attribute dialog. Exportable to VHDL/ Verilog.

**Inputs**

en

If set to 1 the counter is enabled!

C

The clock input. A rising edge increases the counter.

clr

Synchronous reset of the counter if set to 1.

**Outputs**

out

Returns the counted value.

ovf

Overflow output. This pin is set to 1 if the counter is on its maximal value and the en input is set to 1.

**Attributes****Data Bits**

Number of data bits used.

**Inverted inputs**

You can select the inputs that are to be inverted.

**Label**

The name of this element.

**Rotation**

The orientation of the Element in the circuit.

**Use as measurement value**

If set the value is a measurement value and appears in the graph and data table. In addition, a label must be specified that can serve as identification of the value.

**Program Counter**

Makes this register a program counter. The value of this register is returned to the external assembler IDE to mark the current line of code during debugging.



### 11.5. Counter with preset

A counter whose value can be set. In addition, a maximum value and a counting direction can be specified. Exportable to VHDL/Verilog.

**Inputs**

**en**

If set to 1 the counter is enabled!

**C**

The clock input. A rising edge increases or decreases the counter.

**dir**

Specifies the counting direction. A 0 means upwards.

**in**

This data word is stored in the counter when ld is set.

**ld**

If set, the value at input 'in' is stored in the counter at the next clock signal.

**clr**

Synchronous reset of the counter if set to 1.

**Outputs**

**out**

Returns the counted value.

**ovf**

Overflow output. It is set to 1 if the 'en' input is set to 1 and if the counter reaches its maximum value when counting up, or has reached 0 when counting down.

**Attributes****Data Bits**

Number of data bits used.

**Maximum Value**

If a zero is entered, the maximum possible value is used (all bits are one).



**Inverted inputs**

You can select the inputs that are to be inverted.

**Label**

The name of this element.

**Rotation**

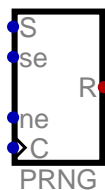
The orientation of the Element in the circuit.

**Use as measurement value**

If set the value is a measurement value and appears in the graph and data table. In addition, a label must be specified that can serve as identification of the value.

**Program Counter**

Makes this register a program counter. The value of this register is returned to the external assembler IDE to mark the current line of code during debugging.



## 11.6. Random Number Generator

Can be used to generate random numbers. When the simulation is started, the generator is reinitialized so that a new pseudo-random number sequence is generated at each start. The generator can be initialized in the running simulation with a defined seed value to generate a defined pseudo-random number sequence.

**Inputs**

**S**

New seed value of the generator.

**se**

If set, the random number generator is reinitialized with the new seed value at the next rising clock edge.

**ne**

If set, a new random number is output at the next rising clock edge.

**C**

The clock input.

**Outputs**

**R**

Output of the pseudorandom number.

**Attributes****Data Bits**

Number of data bits used.

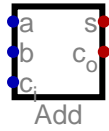
**Label**

The name of this element.

**Rotation**

The orientation of the Element in the circuit.

## 12. Arithmetic



### 12.1. Adder

A component for simple add calculations. Adds the two integer values from input a and input b ( $a+b$ ). The result will be incremented by one if the carry input is set. Exportable to VHDL/Verilog.

#### Inputs

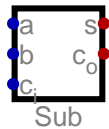
- a  
First input to add.
- b  
Second input to add.
- c\_i  
Carry input, if set the result is incremented by one.

#### Outputs

- s  
The result of the addition
- c\_o  
Carry output. If set there was an overflow.

#### Attributes

- Label  
The name of this element.
- Data Bits  
Number of data bits used.
- Rotation  
The orientation of the Element in the circuit.



### 12.2. Subtract

A component for simple subtractions. Subtracts binary numbers on input a and input b ( $a-b$ ). If the carry input is set to 1 the result is decremented by 1. Exportable to VHDL/Verilog.

### Inputs

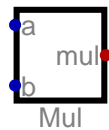
- a  
Input a for subtraction.
- b  
Input b for subtraction.
- c\_i  
Carry input, if set the result is decremented by one.

### Outputs

- s  
Output returns the result of the subtraction.
- c\_o  
Output returns 1 if an overflow occurred.

### Attributes

- Label  
The name of this element.
- Data Bits  
Number of data bits used.
- Rotation  
The orientation of the Element in the circuit.



## 12.3. Multiply

A component for multiplication. Multiplies the integer numbers on input pin a and input pin b. Exportable to VHDL/Verilog.

### Inputs

- a  
Input a for multiplication.
- b  
Input b for multiplication.

### Outputs

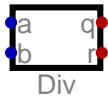
- mul  
Output for the result of the multiplication.

### Attributes

- Label  
The name of this element.
- Signed Operation  
If selected the operation is performed with signed (2th complement) values.
- Data Bits  
Number of data bits used.

**Rotation**

The orientation of the Element in the circuit.

**12.4. Division**

A component for division. Divides the integer applied to input a by the integer applied to input b. If the divisor is zero, it is divided by one instead. In signed division, the remainder is always positive.

**Inputs**

a  
dividend

b  
divisor

**Outputs**

q  
quotient

r  
remainder

**Attributes****Label**

The name of this element.

**Data Bits**

Number of data bits used.

**Signed Operation**

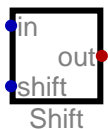
If selected the operation is performed with signed (2th complement) values.

**Remainder always positive**

If set, the remainder of a signed division is always positive.

**Rotation**

The orientation of the Element in the circuit.

**12.5. Barrel shifter**

A component for bit shifting. Shifts the input value by the number of bits given by the shift input.

### Inputs

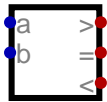
- in  
Input with bits to be shifted.
- shift  
Input with shift width.

### Outputs

- out  
Output with shifted value.

### Attributes

- Label  
The name of this element.
- Data Bits  
Number of data bits used.
- shift input has sign  
shift input data has two complement format
- Direction  
Set direction.
- Mode  
Mode of barrel shifter
- Rotation  
The orientation of the Element in the circuit.



## 12.6. Comparator

A component for comparing bit values. Compares the binary numbers on input pin a and input pin b and sets the corresponding outputs. Exportable to VHDL/Verilog.

### Inputs

- a  
Input a to compare.
- b  
Input b to compare.

### Outputs

- >  
Output is 1 if input a is greater than input b
- =  
Output is 1 if input a equals input b
- <  
Output is 1 if input a is less than input b

### Attributes

- Label  
The name of this element.

**Data Bits**

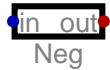
Number of data bits used.

**Signed Operation**

If selected the operation is performed with signed (2th complement) values.

**Rotation**

The orientation of the Element in the circuit.

**12.7. Negation**

Negation in the 2th complement Exportable to VHDL/Verilog.

**Inputs**

in

Input of the data word to be negated in 2th complement

**Outputs**

out

Returns the result of the negation in 2th complement.

**Attributes****Data Bits**

Number of data bits used.

**Rotation**

The orientation of the Element in the circuit.

**12.8. Sign extender**

Increases the bit width of a signed value keeping the values sign. If the input is a single bit, this bit will be output on all output bits. Exportable to VHDL/Verilog.

**Inputs**

in

Input value. The input bit width must be smaller than the output bit width!

**Outputs**

out

Extended input value. The input bit width must be smaller than the output bit width!

**Attributes****Label**

The name of this element.

**Input Bit Width**

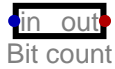
The number of output bits must be greater than the number of input bits.

**Output Bit Width**

The number of output bits must be greater than the number of input bits.

**Rotation**

The orientation of the Element in the circuit.

**12.9. Bit counter**

Returns the number of 1-bits in the input value.

**Inputs**

in

The input which 1-bits are counted.

**Outputs**

out

Outputs the number of 1-bits.

**Attributes****Data Bits**

Number of data bits used.

**Rotation**

The orientation of the Element in the circuit.

**13. Switches****13.1. Switch**

Simple switch. There is no gate delay: A signal change is propagated immediately.

**Outputs**

A1

One of the connections of the switch.

B1

One of the connections of the switch.

**Attributes****Data Bits**

Number of data bits used.

**Label**

The name of this element.

**Pole count**

Number of poles available.

**Closed**

Sets the initial state of the switch.

**Rotation**

The orientation of the Element in the circuit.

**Mirror**

Mirrors the component in the circuit.

**Switch behaves like an input**

If the model is analyzed, the switch behaves like an input, where "open" corresponds to '0' and "closed" to '1'.



### 13.2. Double Throw Switch

Double Throw Switch. There is no gate delay: A signal change is propagated immediately.

**Outputs****A1**

One of the connections of the switch.

**B1**

One of the connections of the switch.

**C1**

One of the connections of the switch.

**Attributes****Data Bits**

Number of data bits used.

**Label**

The name of this element.

**Pole count**

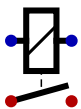
Number of poles available.

**Rotation**

The orientation of the Element in the circuit.

**Mirror**

Mirrors the component in the circuit.



### 13.3. Relay

A relay is a switch which can be controlled by a coil. If a current flows through the coil, the switch is closed or opened. There is no flyback diode so the current direction does not matter. The switch is actuated if the inputs have different values. The relay behaves similar to an XOr gate.



### Inputs

in1

One of the inputs to control the relay.

in2

One of the inputs to control the relay.

### Outputs

A1

One of the connections of the switch.

B1

One of the connections of the switch.

### Attributes

Data Bits

Number of data bits used.

Label

The name of this element.

Pole count

Number of poles available.

Relay is normally closed.

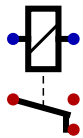
If set the relay is closed if the input is low.

Rotation

The orientation of the Element in the circuit.

Mirror

Mirrors the component in the circuit.



## 13.4. Double Throw Relay

A relay is a switch which can be controlled by a coil. If a current flows through the coil, the switch is closed or opened. There is no flyback diode so the current direction does not matter. The switch is actuated if the inputs have different values. The relay behaves similar to an XOr gate.

### Inputs

in1

One of the inputs to control the relay.

in2

One of the inputs to control the relay.

## Outputs

- A1  
One of the connections of the switch.
- B1  
One of the connections of the switch.
- C1  
One of the connections of the switch.

## Attributes

- Data Bits  
Number of data bits used.
- Label  
The name of this element.
- Pole count  
Number of poles available.
- Rotation  
The orientation of the Element in the circuit.
- Mirror  
Mirrors the component in the circuit.



## 13.5. P-Channel FET

P-Channel Field Effect Transistor. The bulk is connected to the pos. voltage rail and the transistor is simulated without a body diode.

### Inputs

- G  
Gate

### Outputs

- S  
Source
- D  
Drain

### Attributes

- Data Bits  
Number of data bits used.
- Unidirectional  
Unidirectional transistors propagate a signal only from source to drain. They are much faster to simulate than bidirectional transistors. Since there is no feedback from drain to source, in this mode, the transistor can not short the connected wires when it is conducting. Thus, this mode is necessary to simulate certain CMOS circuits.
- Label  
The name of this element.

Rotation

The orientation of the Element in the circuit.

Mirror

Mirrors the component in the circuit.



### 13.6. N-Channel FET

N-Channel Field Effect Transistor. The bulk is connected to ground and the transistor is simulated without a body diode.

Inputs

G

Gate

Outputs

D

Drain

S

Source

Attributes

Data Bits

Number of data bits used.

Unidirectional

Unidirectional transistors propagate a signal only from source to drain. They are much faster to simulate than bidirectional transistors. Since there is no feedback from drain to source, in this mode, the transistor can not short the connected wires when it is conducting. Thus, this mode is necessary to simulate certain CMOS circuits.

Label

The name of this element.

Rotation

The orientation of the Element in the circuit.

Mirror

Mirrors the component in the circuit.



### 13.7. Fuse

A fuse used to build a one time programmable memory.

Outputs

out1

One of the connections of the switch.

out2

One of the connections of the switch.

#### Attributes

##### Programmed

If set a diode is "blown" or "programmed". At a floating gate FET the floating gate is charged. You can change this setting with the [P] key.

##### Rotation

The orientation of the Element in the circuit.



### 13.8. Diode to VDD

A simplified unidirectional diode, used to pull a wire to VDD. It is used to implement a wired OR. So it is necessary to connect a pull down resistor to the diodes output. In the simulation the diode behaves like an active gate with a trivalent truth table: If the input high, the output is also high. In all other cases (input is low or high z) the output is in high z state. So two anti parallel connected diodes can keep each other in high state, which is not possible with real diodes. This is an ideal diode: There is no voltage drop across a forward-biased diode.

#### Inputs

in

If the input is high also the output is high. In all other cases the output is in high z state.

#### Outputs

out

If the input is high also the output is high. In all other cases the output is in high z state.

#### Attributes

##### Programmed

If set a diode is "blown" or "programmed". At a floating gate FET the floating gate is charged. You can change this setting with the [P] key.

##### Rotation

The orientation of the Element in the circuit.



### 13.9. Diode to Ground

A simplified unidirectional diode, used to pull a wire to ground. It is used to implement a wired AND. So it is necessary to connect a pull up resistor to the diodes output. If the input low, the output is also low. In the other cases (input is high or high z) the output is in high z state. So two anti parallel connected diodes can keep each other in low state, which is not possible with real diodes. So this is a ideal diode: There is no voltage drop across a forward-biased diode.

**Inputs**

in

If the input is low also the output is low. In all other cases the output is in high z state.

**Outputs**

out

If the input is low also the output is low. In all other cases the output is in high z state.

**Attributes****Programmed**

If set a diode is "blown" or "programmed". At a floating gate FET the floating gate is charged. You can change this setting with the [P] key.

**Rotation**

The orientation of the Element in the circuit.

**13.10. P-Channel floating gate FET**

P-Channel Floating Gate Field Effect Transistor. The bulk is connected to ground and the transistor is simulated without a body diode. If there is a charge stored in the floating gate, the fet isn't conducting even if the gate is low.

**Inputs**

G

Gate

**Outputs**

S

Source

D

Drain

**Attributes****Data Bits**

Number of data bits used.

**Label**

The name of this element.

**Programmed**

If set a diode is "blown" or "programmed". At a floating gate FET the floating gate is charged. You can change this setting with the [P] key.

**Rotation**

The orientation of the Element in the circuit.

**Mirror**

Mirrors the component in the circuit.



### 13.11. N-Channel floating gate FET

N-Channel Floating Gate Field Effect Transistor. The bulk is connected to ground and the transistor is simulated without a body diode. If there is a charge stored in the floating gate, the fet isn't conducting even if the gate is high.

Inputs

G  
Gate

Outputs

D  
Drain  
S  
Source

Attributes

Data Bits

Number of data bits used.

Label

The name of this element.

Programmed

If set a diode is "blown" or "programmed". At a floating gate FET the floating gate is charged. You can change this setting with the [P] key.

Rotation

The orientation of the Element in the circuit.

Mirror

Mirrors the component in the circuit.



### 13.12. Transmission-Gate

A real transmission-gate is build from only two transistors. Therefore, it is often used to save transistors during implementation on silicon.

Inputs

S  
control input.  
 $\neg S$   
inverted control input

#### Outputs

- A  
input A
- B  
input B

#### Attributes

- Data Bits  
Number of data bits used.
- Rotation  
The orientation of the Element in the circuit.

## 14. Misc.

Test

### 14.1. Test case

Describes a test case. In a test case you can describe how a circuit should behave. It can then be automatically checked whether the behavior of the circuit actually corresponds to this description. If this is not the case, an error message is shown. The help text of the test case editor describes in detail how such a test case can be created. Exportable to VHDL/Verilog.

#### Attributes

- Label  
The name of this element.
- Test data  
The description of the test case. Details of the syntax can be found in the help dialog of the test data editor.
- Enabled  
Enables or disables this component.

## 15. Misc. - Decoration

### Text

#### 15.1. Text

Shows a text in the circuit. Does not affect the simulation. The text can be changed in the attribute dialog.

#### Attributes

- Description  
A short description of this element and its usage.
- Font Size  
Sets the font size to use for this text.

**Rotation**

The orientation of the Element in the circuit.

**Orientation**

Position of the coordinate relative to the text.

**Snap To Grid**

If set, the component is aligned with the grid.

**Text**

## 15.2. Rectangle

Shows a rectangle in the circuit. Does not affect the simulation. If a minus sign is used as the heading, the heading is omitted.

**Attributes****Label**

The name of this element.

**Width**

Width in grid units

**Height**

Height in grid units

**Font Size**

Sets the font size to use for this text.

**Text Inside**

Place text inside the rectangle.

**Text at the bottom**

Place text at the bottom of the rectangle.

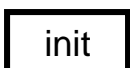
**Text on the right**

Place text to the right of the rectangle.

**Snap To Grid**

If set, the component is aligned with the grid.

## 16. Misc. - Generic



### 16.1. Generic Initialization

Code that is executed to start a generic circuit directly. If a generic circuit is to be started directly, such a component must be present. Exportable to VHDL/Verilog.

**Attributes****Label**

The name of this element.

**Enabled**

Enables or disables this component.



## Generic Parameterization

Statements used to generify a circuit.

## Code

### 16.2. Code

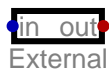
Code that is executed when a generic circuit is made concrete. Can be used, for example, to add components or wires to a circuit. Exportable to VHDL/Verilog.

#### Attributes

## Generic Parameterization

Statements used to generify a circuit.

## 17. Misc. - VHDL/Verilog



### 17.1. External

Component to execute an external process to calculate the logic function. Is used to specify the behaviour of a component by VHDL or Verilog. The actual simulation of the behavior must be done with an external simulator. At present only the VHDL simulator ghdl and the verilog simulator Icarus Verilog are supported. The label of the component must match the name of the entity or module! Exportable to VHDL/Verilog.

#### Inputs

in

#### Outputs

out

#### Attributes

##### Label

The name of this element.

##### Width

Width of symbol if this circuit is used as an component in an other circuit.

##### Inputs

The inputs of the external process. It is a comma-separated list of signal names. For each signal name, a number of bits separated by a colon can be specified. The inputs of an 8-bit adder could thus be described as "a:8,b:8,c\_in".

##### Outputs

The outputs of the external process. It is a comma-separated list of signal names. For each signal name, a number of bits separated by a colon can be specified. The outputs of an 8-bit adder could thus be described as "s:8,c\_out".

##### Program code

The program code to be executed by the external application.

**Application**

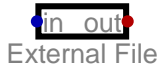
Defines which application to use.

**GHDH Options**

Options that are used for all processing steps by GHDH.

**IVerilog Options**

Options that are used for all processing steps by IVerilog.

**17.2. External File**

Component to execute an external process to calculate the logic function. Is used to specify the behaviour of a component by VHDL or Verilog. The actual simulation of the behavior must be done with an external simulator. At present only the VHDL simulator ghdl and the verilog simulator Icarus Verilog are supported. The label of the component must match the name of the entity or module! Exportable to VHDL/Verilog.

**Inputs**

in

**Outputs**

out

**Attributes****Label**

The name of this element.

**Width**

Width of symbol if this circuit is used as an component in an other circuit.

**Inputs**

The inputs of the external process. It is a comma-separated list of signal names. For each signal name, a number of bits separated by a colon can be specified. The inputs of an 8-bit adder could thus be described as "a:8,b:8,c\_in".

**Outputs**

The outputs of the external process. It is a comma-separated list of signal names. For each signal name, a number of bits separated by a colon can be specified. The outputs of an 8-bit adder could thus be described as "s:8,c\_out".

**Program code**

The file containing the program code to be executed by the external application.

**Application**

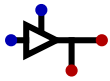
Defines which application to use.

**GHDH Options**

Options that are used for all processing steps by GHDH.

**IVerilog Options**

Options that are used for all processing steps by IVerilog.



### 17.3. Pin Control

Control logic for a bidirectional pin. This component is necessary only in the context of VHDL or Verilog generation, in order to create a bidirectional HDL port! If you don't want to use a bidirectional IO-port on an FPGA, don't use this component! The PinControl component cannot be used in an embedded circuit! It is only allowed at the top level circuit! Exportable to VHDL/Verilog.

#### Inputs

- wr  
The data to be output.
- oe  
Activates the output.

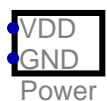
#### Outputs

- rd  
The data to be read.
- pin  
The connector for the actual pin. Only a single output should be connected here.

#### Attributes

- Data Bits  
Number of data bits used.
- Rotation  
The orientation of the Element in the circuit.
- Mirror  
Mirrors the component in the circuit.

## 18. Misc.



### 18.1. Power

Has no function. Makes sure that VDD and GND are connected. Can be used in 74xx circuits to generate the pins for the voltage supply, which are tested for correct wiring.

#### Inputs

- VDD  
Must be connected to VDD!
- GND  
Must be connected to GND!

#### Attributes

**Label**

The name of this element.

**Rotation**

The orientation of the Element in the circuit.

**18.2. Bidirectional Splitter**

Can be used for data buses and simplifies especially the construction of memory modules in a DIL package, as the implementation of the data bus is simplified.

**Inputs****OE**

When set, the value at the common data terminal D is output to the bit outputs D[i], if not, the bits D[i] are output to the common output D.

**Outputs****D**

The common data connection.

**D0**

The data bit 0 of the bus splitter.

**Attributes****Data Bits**

Number of data bits used.

**Rotation**

The orientation of the Element in the circuit.

**Spreading**

Configures the spread of the inputs and outputs in the circuit.

**18.3. Reset**

The output of this component is held high during the initialisation of the circuit. After the circuit has stabilized the output goes to low. If the output is inverted it behaves the opposite way. Exportable to VHDL/Verilog.

**Outputs****Reset**

Reset Output.

**Attributes****Label**

The name of this element.

**Inverted output**

If selected the output is inverted.

**Rotation**

The orientation of the Element in the circuit.



### 18.4. Break

If this component is used in the circuit, the "Run To Break" button between "Start" and "Stop" is enabled. This button clocks the circuit until a rising edge on the input of this component is detected. This element can be used for debugging by clocking the circuit to any breakpoint. Also an assembler command BRK can be implemented. This allows to execute a program up to the next BRK command. This function can only be used if the real-time clock is deactivated!

**Inputs****brk**

Stops the fast simulation clocking if a rising edge is detected.

**Attributes****Label**

The name of this element.

**Enabled**

Enables or disables this component.

**Timeout cycles**

If this amount of cycles is reached without a break signal, an error is created.

**Rotation**

The orientation of the Element in the circuit.



### 18.5. Stop

A rising edge at the input stops the simulation. Has the same effect as pressing the Stop button in the toolbar.

**Inputs****stop**

A rising edge stops the simulation.

**Attributes****Label**

The name of this element.

**Inverted inputs**

You can select the inputs that are to be inverted.

**Rotation**

The orientation of the Element in the circuit.

Async

## 18.6. Asynchronous Timing

Allows configuration of the timing of an asynchronous sequential circuit such as a Muller-pipeline. The circuit must be started in single gate step mode and must be able to reach a stable state at startup. The sequential circuit can then be started interactively or with a reset gate. It is not allowed to use a regular clock component in this mode.

### Attributes

- Start real time clock

  - If enabled the runtime clock is started when the circuit is started

- Frequency/Hz

  - The real time frequency used for the real time clock

## E Library

- 27c801:** 8 Mbit (1Mb x 8) UV EPROM
- 28c010:** 1-Megabit (128K x 8) Paged Parallel EEPROM; DATA Polling for End of Write Detection not implemented!
- 28c16:** 16K (2K x 8) Parallel EEPROM; DATA Polling for End of Write Detection not implemented!
- 28c64:** 64K (8K x 8) Parallel EEPROM; DATA Polling for End of Write Detection not implemented!
- 28c256:** 256K (32K x 8) Paged Parallel EEPROM; DATA Polling for End of Write Detection not implemented!
- 28c512:** 512K-Bit (64K x 8) CMOS Parallel EEPROM; DATA Polling for End of Write Detection not implemented!
- 7400:** quad 2-input NAND gate
- 7401:** quad 2-input NAND gate with open-collector outputs
- 7402:** quad 2-input NOR gate
- 7403:** quad 2-input NAND gate with open-collector outputs, different pinout than 7401
- 7404:** hex inverter
- 7405:** hex inverter, open-collector output
- 7406:** hex inverter buffer, open-collector output
- 7407:** hex buffer, open-collector output
- 7408:** quad 2-input AND gate
- 7409:** quad 2-input AND gate with open-collector outputs
- 7410:** triple 3-input NAND gate
- 7411:** triple 3-input AND gate
- 7412:** triple 3-input NAND gate with open-collector outputs
- 7413:** dual 4-input NAND gate, Schmitt trigger
- 7414:** hex inverter, Schmitt trigger
- 7415:** triple 3-input AND gate with open-collector outputs
- 7416:** hex inverter buffer, open-collector output, same as 7406
- 7417:** hex buffer, open-collector output, same as 7407
- 7420:** dual 4-input NAND gate
- 7421:** dual 4-input AND gate
- 7425:** dual 4-input NOR gate
- 7427:** triple 3-input NOR gate
- 7428:** quad 2-input NOR buffer
- 7430:** 8-input NAND gate
- 7432:** quad 2-input OR gate
- 7440:** dual 4-input NAND buffer
- 7442:** 4-line BCD to 10-line decimal decoder
- 7447:** BCD to 7-segment decoder, active low
- 7448:** BCD to 7-segment decoder, active high
- 7451:** 2-input/3-input AND-NOR gate
- 7454:** 2-3-2-3-line AND NOR gate
- 7455:** 2 wide 4-input AND-NOR gate
- 7458:** dual AND OR gate
- 7474:** dual D-flip-flop
- 7476:** dual J-K flip-flops with preset and clear
- 7480:** Gated Full Adder with Complementary Inputs and Complementary Sum Outputs
- 7482:** 2-bit binary full adder

**7483:** 4-bit binary full adder  
**7483Real:** 4-bit binary full adder, real gates  
**7485:** 4-bit comparator  
**7486:** quad 2-input XOR gate  
**7489:** 64-bit RAM  
**74107:** dual J-K flip-flops with clear  
**74109:** Dual J-NOT-K flip-flop with set and reset; positive-edge-trigger  
**74112:** Dual J-K negative-edge-triggered flip-flop, clear and preset  
**74116:** dual 4-bit D-type latches  
**74133:** 13-input NAND gate  
**74138:** 3-line to 8-line decoder/demultiplexer, inverted out  
**74139:** dual 2-line to 4-line decoder/demultiplexer  
**74147:** 10-line to 4-line priority encoder  
**74148:** 8-line to 3-Line priority encoder  
**74150:** 4-line to 16-line data selectors/multiplexers  
**74151:** 3-line to 8-line data selectors/multiplexers  
**74153:** dual 4-line to 1-line data selectors/multiplexers  
**74154:** 4-line to 16-line decoders/demultiplexers  
**74157:** quad 2-line to 1-line data selectors/multiplexers  
**74160:** decimal synchronous counter, async clear  
**74161:** hex synchronous counter, async clear  
**74162:** decimal synchronous counter  
**74162Real:** decimal synchronous counter, real gates  
**74163:** hex synchronous counter  
**74164:** 8-bit parallel-out serial shift register, asynchronous clear  
**74165:** parallel-load 8-bit shift register  
**74166:** 8-Bit Parallel-In/Serial-Out Shift Register  
**74173:** quad 3-state D flip-flop with common clock and reset  
**74174:** hex D-flip-flop  
**74181:** 4-bit arithmetic logic unit  
**74182:** look-ahead carry generator  
**74189:** 64-Bit Random Access Memory with 3-STATE Outputs  
**74190:** Presettable synchronous 4-bit bcd up/down counter  
**74191:** Presettable synchronous 4-bit binary up/down counter  
**74193:** Synchronous 4-Bit Up/Down Binary Counter with Dual Clock  
**74198:** 8-bit shift register  
**74238:** 3-line to 8-line decoder/demultiplexer  
**74244:** octal 3-state buffer/line driver/line receiver  
**74245:** octal bus transceivers with 3-state outputs  
**74247:** BCD to 7-segment decoder, active low, tails on 6 and 9  
**74248:** BCD to 7-segment decoder, active high, tails on 6 and 9  
**74253:** dual tri state 4-line to 1-line data selectors/multiplexers  
**74260:** dual 5-input NOR gate  
**74266:** quad 2-input XNOR gate  
**74273:** octal D-type flip-flop with clear  
**74280:** 9 bit Odd-Even Parity Generator-Checker  
**74283:** 4-bit binary full adder, alternative pinning  
**74299:** 8-Input Universal Shift/Storage Register with Common Parallel I/O Pins  
**74373:** octal transparent latches  
**74374:** octal positive-edge-triggered flip-flops



**74377:** Octal D Flip-Flop with enable  
**74382:** 4-Bit Arithmetic Logic Unit  
**74540:** octal buffer/line driver, inverted  
**74541:** octal buffer/line driver  
**74573:** octal transparent latches, different pinout compared to 74373  
**74574:** octal positive-edge-triggered flip-flops, different pinout compared to 74374  
**74590:** 8-bit binary counter with tri-state output registers  
**74595:** 8-Bit Shift Registers with 3-State Output Registers  
**74670:** 3-state 4-by-4 Register File  
**74682:** 8-bit digital comparator  
**74688:** 8-bit identity comparator  
**74779:** 8-Bit Bidirectional Binary Counter with 3-STATE Outputs  
**74804:** hex 2-input NAND gate <https://www.ti.com/lit/ds/symlink/sn74as804b.pdf>  
**74805:** hex 2-input NOR gate <http://www.ti.com/lit/ds/symlink/sn54as805b.pdf>  
**74808:** hex 2-input AND gate <http://www.ti.com/lit/ds/symlink/sn54as808b.pdf>  
**74832:** hex 2-input OR gate <http://www.ti.com/lit/ds/symlink/sn54as832b.pdf>  
**744017:** Johnson decade counter with 10 decoded outputs  
**744075:** triple 3-input OR gate  
**A623308A:** 8K X 8 BIT CMOS SRAM  
**RAM32Bit:** Ein 32-Bit-Speicher, der Bytezugriff ermöglicht und auch mit nicht ausgerichteten Speicheradressen umgehen kann.