12

Editor Reference

12.1 Basic Structures

In this section we use syntax diagrams to define the structure of user input as for example schematic names, parameters of macro cells, width of wires etc. We will show the corresponding syntax diagrams in the different sections. There are some basic syntactical structures, on which these syntax diagrams are based. We will describe these constructs in the following introduction.

The following basic symbols of the underlying programming language are defined:

`A ... Z, a ... z`	letter
`0 1 2 3 4 5 6 7 8 9`	digit
`+ - * / ^ %`	arithmetical operators
`< <= >= > == !=`	compare operators
`( )`	brackets in expressions
`[ ]`	brackets for parameter lists
`,`	split symbol in parameter lists
`@`	introduction of a wire variable

The following standard functions are defined:

`log`	logarithm to base 2
`upper`	next larger integer
`lower`	next smaller integer
`sqrt`	square root
`max`	maximum of two numbers
`min`	minimum of two numbers

With these basic elements we define the following structures which are essential for the syntax diagrams in the next few paragraphs.

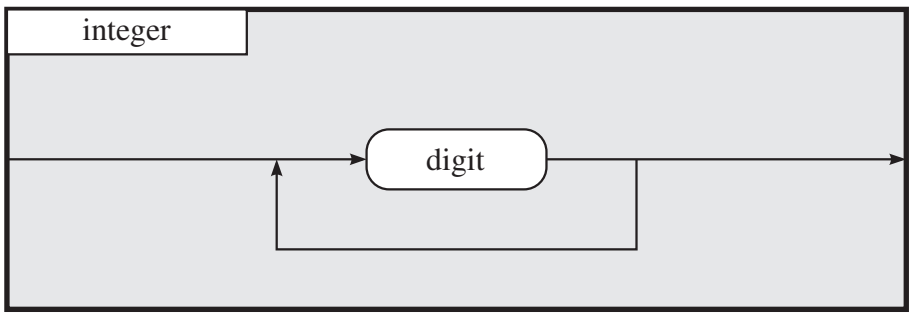

Figure 12.1: Syntactical structure of an unsigned integer

The diagram in figure 12.1 specifies the structure of an unsigned integer. From this diagram follows that an unsigned integer may be an arbitrary sequence of digits, but on a real machine an integer n must be in the range

`0 <=` n `<= 2147483648`

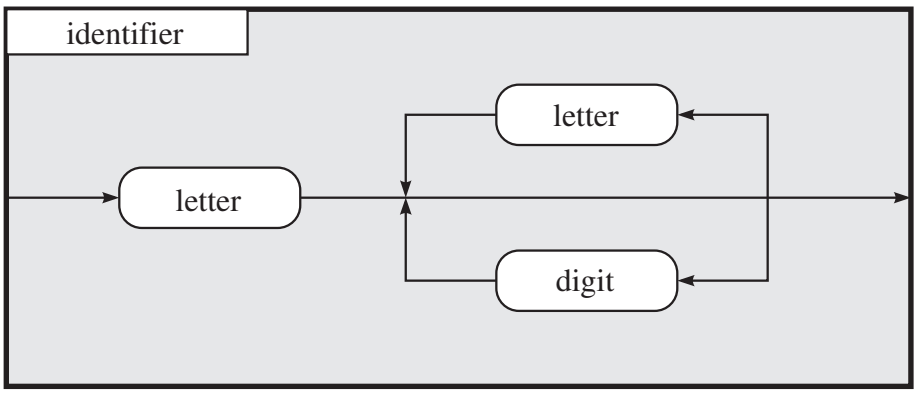

Figure 12.2: Syntactical structure of identifiers

A second important structure is the term identifier for which the diagram

is given in figure 12.2. An identifier may be an arbitrary sequence of letters and digits preceeded by a letter. Theoretically an identifier may have any length but in most cases the system will induce certain restrictions. For example the length of a schematic name must not exceed 80 characters. There are some other inputs where we have other restrictions to the length.

In the diagrams 12.1 and 12.2 the rounded elements represent elementary objects. In the following syntax diagrams there will be elements specified by rectangles. They represent objects which themselves are given by syntax diagrams.

12.2 Schematics

12.2.1 Load Schematic

This menupoint is used to load an existing schematic as well as to create a new one. If you select the entry `Load` from the `-Schematics-` submenu a list window will popup in the workarea. This window contains the list of schematics in the directory given by the environment variable `DAGDIR`. The list is in alphabetical order and it contains one special entry at its top which is called `**** New ****`. You must select this entry if you want to create a new schematic sheet, as it is shown in figure 12.3.

After the window is popped up you are in the name selection mode. When you move the pointer near to an entry in the schematic list it will be highlighted. You can notice that the colour of the schematic name changes from white to black and now is displayed on the background of a white bar. Move the pointer to the next entry in the list and you will notice the change of the highlighted item. Of course you cannot watch this behaviour if you have just started a new session of the editor where still no schematics have been created. In this case the only entry in the list is `**** New ****`.

The size of the list window is fixed, i.e. it does not depend on the number of items in `DAGDIR`. If there are more schematics in `DAGDIR` than can be displayed in the list window you can scroll the list up and down. You can

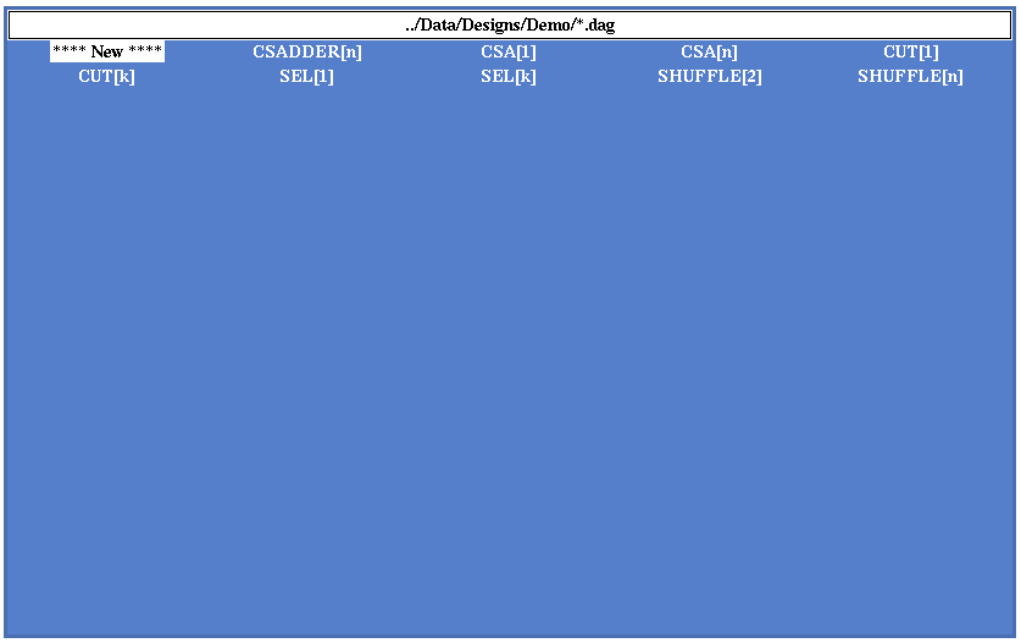

Figure 12.3: List window for schematic names in `DAGDIR` with selected entry `**** New ****`

scroll up the list by pressing the middle mouse button in the lower half of the list window. Note that you can repeatedly scroll up and the list window will be empty if you have scrolled beyond the bottom of the list . In the same way you can scroll down when you press the middle mouse button within the upper half of the list window until you reach the top of the list. The editor keeps the current scroll position in mind, so that further schematic selections are based on the same list offset. If you do not see the `**** New ****` entry the list is scrolled up, move the pointer in the upper half of the window and press the middle mouse button repeatedly until the entry appears in the window.

If you select the entry `**** New ****` an input window is popped up in the workarea with the following prompt (c.f. figure 12.4)

```
Please enter new Schematic Name:  <          >
```

The pointer has automatically been moved into this window and you can type in the name for a new schematic to be created. The distance between

the angle brackets < > indicates the possible length of a schematic name. This name must not be longer than 80 characters and the input routine does not allow you to type in more than this number.

Figure 12.4: Input window for the specification of a new schematic name

When you type in a schematic name you have to follow the syntactical rules, i.e. not every arbitrary sequence of characters is a correct schematic name. The following diagram in figure 12.5 shows the syntax of schematic names.

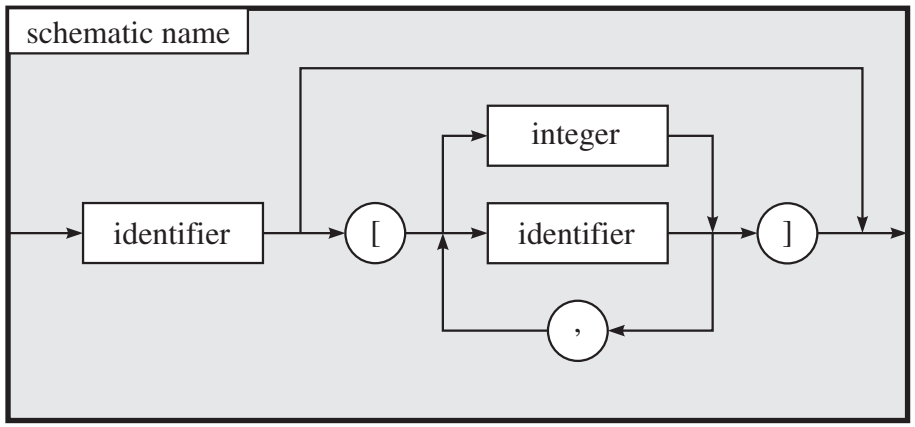

Figure 12.5: Syntax diagram for schematic names

A schematic name is given by an identifier and an optional list of parameters. This list of parameters may contain identifiers as well as integers. The diagram implies that you may use an infinite list of parameters, but in the editor you may only use up to 32 parameters. The parameters of the type identifier are called free, because they may be substituted by any non negative natural number. The other parameters are called fixed. All free parameters of a schematic may be used to create expressions for parameterized macro cells or the width of wires. This will be explained in detail in section 12.3.1.

Some examples for legal schematic names are the following:

```
CSADDER[n]
SEL[1]
NGrid[n,d1,d2,d3]
Matrix[rows,columns]
```

Illegal schematic names are the following:

`SHUFFLE[n`	parameter list not closed
`1CUT[k]`	name is not an identifier
`Matrix[-1,0]`	first parameter is not a non negative integer
`NGrid[,d1,d2]`	parameter list must not begin with a comma
`CSA[n/2]`	parameter must not be an arithmetical expression

If you type in a schematic name and press the return key the name will be analyzed by a parser. In the case that the name is syntactically not correct the system will display an error message in the message window below the workarea:

```
!  Error !  New Name "SHUFFLE[n" is syntactical incorrect
```

The input window will popup again, so that you can retry to enter the schematic name. If you made a mistake and do not want to enter a schematic name, you can cancel the input routine by simply pressing the return key to the empty input window.

If the name of the schematic is syntactically correct, the system will check, whether the new name is a redefinition of an existing schematic. You must not define two general equations with the same schematic identifier. If you have already defined a schematic `CSA[n]`, you are not allowed to define a second general equation of the form `CSA[k]`. In this case the system will respond with the error message

```
!  Error !  New Name "CSA[k]" is Redefinition of CSA[n]
```

If you want to redefine a schematic with new parameters, you have to delete all conflicting schematics .

⇝ 12.2.5

If there is no conflict with other schematic definitions, the input window will disappear and the system will display the message

```
New Schematic "CSA[n]" opened
```

in the message window below the workarea. Note: If you have selected the entry `**** New ****` from the schematic list, but typed in the name of an existing schematic, the corresponding drawing will be shown in the workarea (you not really opened a new schematic in this case).

The name of the schematic is displayed in the upper right corner of the graphical surface in one of the panels of the environment window. In the workarea you will see an empty white frame which represents the border of the schematic. Within this frame you have to place all elements of the schematic like macros, basic cells, wires etc.

12.2.2 Save Schematic

If you select the entry `Save` from the `-Schematics-` submenu the current schematic is saved to a file in the directory given by the environment variable `DAGDIR`. The file is named

```
<schematicname>.dag
```

The file contains the elements needed for the graphical representation of the schematic. These are position, orientation and names of the basic cells and macros, the wires, the comment texts and the additional equations. Each schematic is represented by a graph data structure, i.e. the borders of the schematic or the macros as well as the wires are the edges of this graph. The corners of the schematic and the cells, the wire points and the connections of wires to the cells and the border of the schematic are the nodes of this graph.

After the schematic has been saved to `DAGDIR` a macro cell will be generated, if the schematic has no or only fixed parameters. Such a macro is called a discrete macro, because its elements do not depend on any free parameters. The discrete macros can be selected as cells (menu entry `Enter` from `-Cells-` submenu) from the second cell list window.

Note: you need not to create schematics only with the editor. You can write programs yourself to create correct files in `DAGDIR`. This can be done by using the functions from the basic library for the graph data structure.

12.2.3 Save Schematic under New Name

You can save the current schematic under a new name with the entry `Save As` from the `-Schematics-` submenu. After selecting this point the input window will popup and you can type in a new name for the current schematic at the following prompt:

```
Please enter new Schematic-Name:<        >
```

For the new schematic name you have to consider the syntactical rules
⇝ 12.2.1 defined by diagram 12.5 from section 12.2.1. If you have typed in a name which is not syntactically correct, the input window will popup again and in the message window you can read a message like

```
!  Error !  New Name "SHUFFLE[n" is syntactically incorrect
```

You can enter a new name for the schematic or abort the save function by pressing the return key to an empty input window.

⇝ 12.2.2 When the schematic is saved to `DAGDIR` a macro cell with the name of the new schematic will be generated in the case of discrete macros.

12.2.4 Clear Schematic

With the help of the function `Clear` from the `-Schematics-` submenu you can erase all elements (macros, wires, comments and equations) within a schematic. It clears your workarea and leaves the empty schematic frame. If you have made mistakes during the drawing this function is a fast method to erase all elements.

This function gives an alternative method to create a new schematic. You can load an existing schematic, clear it and save it under a new name. The you have to load the newly created schematic in order not to overwrite the currently loaded schematic.

Note: there is no security check, when you call this function. If you called this function and did not want to clear the schematic, just load the schematic again without saving it. The old drawing will be restored.

12.2.5 Delete Schematic

With the function `Delete` from the `-Schematic-` submenu you can delete a schematic file in `DAGDIR`. After the selection of this entry the schematic list window will popup and you can select the name of a schematic. Note that the `**** New ****` entry does not appear in the list and the order of the schematic names is shifted by one to the left. If you press the left mouse button the current highlighted schematic is deleted. For the corresponding file in `DAGDIR` the command

```
> rm -f <schematicname>.dag
```

is executed. The deletion of the file can be noticed in the change of the schematic list where the name of the just deleted schematic will disappear. Then you can select another schematic for deletion or you can abort the function by pressing the right mouse button within the list window.

Note: be careful with this function. In the current implementation there is no security check which asks you, whether you really want to delete the schematic.

12.3 Cells and Macros

12.3.1 Enter Cells

With the function `Enter` from the `-Cells-` submenu you can select cells and macros and place them within the schematic frame. The cells are divided into the following three groups according to their functionality:

- *Basic cells* are elementary components which compute basic operations. The user can create a basic cell library with the help of a cell editor tool. With this editor it is very easy to make a cell library of

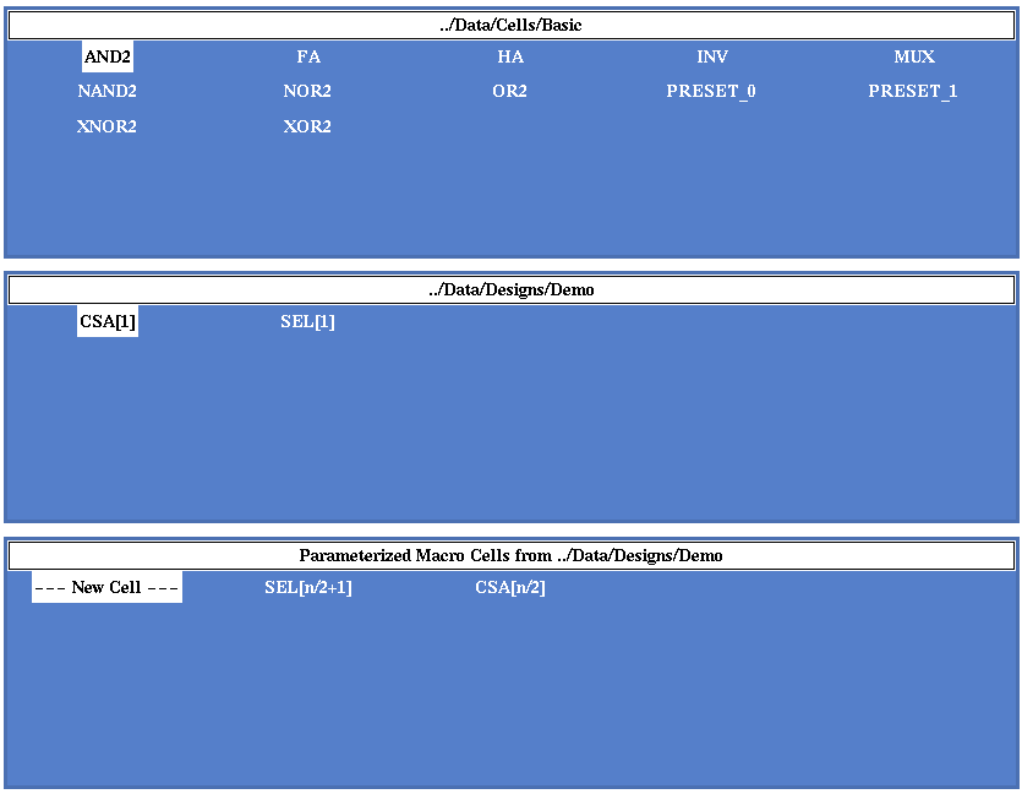

Figure 12.6: Cell selection windows for basic cells, discrete and parameterized macros

commercial design systems available for the CADIC system. Within the current distribution of the CADIC system there are some basic cell libraries of commercial design system contained.

- *Discrete macro cells* are components which are defined by the graphical input of a corresponding schematic. Discrete means that this macro does not depend on free parameters, i.e. the schematic name only contains integer parameters or it has no parameter at all. With the use of such a macro the designer creates a new hierarchy level in the design specification. The wires connected to the border of this schematic will appear as the pins of the graphical macro representation.

- *Parameterized macro cells* are components, which depend on different parameters of the current schematic. These macros can be used espe-

cially within recursive specifications, where you use an instance of the currently specified schematic. Parameterized macro cells can be compared with forward declarations known from programming languages.

Each of these cell groups is displayed in a separate window which is popped up upon the workarea (c.f. figure 12.6). The windows are arranged according to the listing above. In the upmost window the names of the available cells from the basic cell library are displayed, the second window contains the names of the defined discrete macro cells and in the third window the parameterized macro cells are listed, which have been defined during the current editor session.

If you move the pointer into one of the three windows near to the name of a cell or macro, the corresponding entry will be highlighted. Now you can select this item by pressing the left mouse button. After this a graphical representation of the cell or macro is displayed within the workarea. This representation depends on the type of the cell.

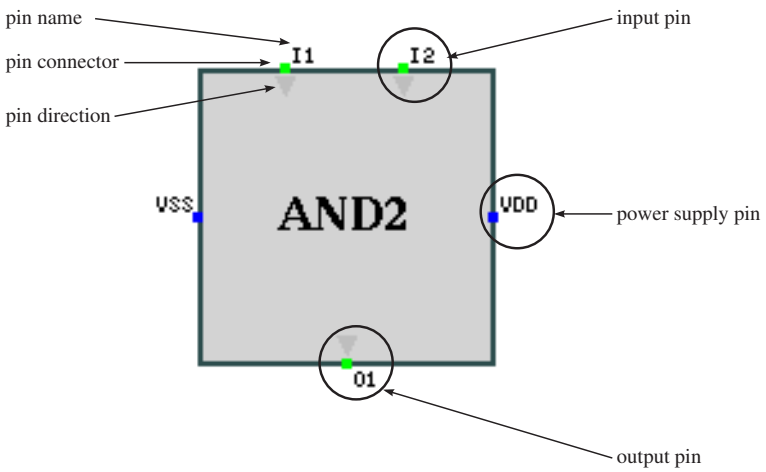

Figure 12.7: Graphical representation of an and gate with two inputs selected from the basic cell library

Figure 12.7 shows the typical layout of a basic cell. The displayed information is called the interface of the cell. It contains the size of the cell, the names and positions and directions of its pins. As shown in figure 12.7 we distinguish between signal and power supply pins. During an editor ses-

sion normallay only the signal pins are used and connected to wires. The power supply pins are left open, because the supply nets will be generated automatically by an appropriate tool (c.f. chapter 8).

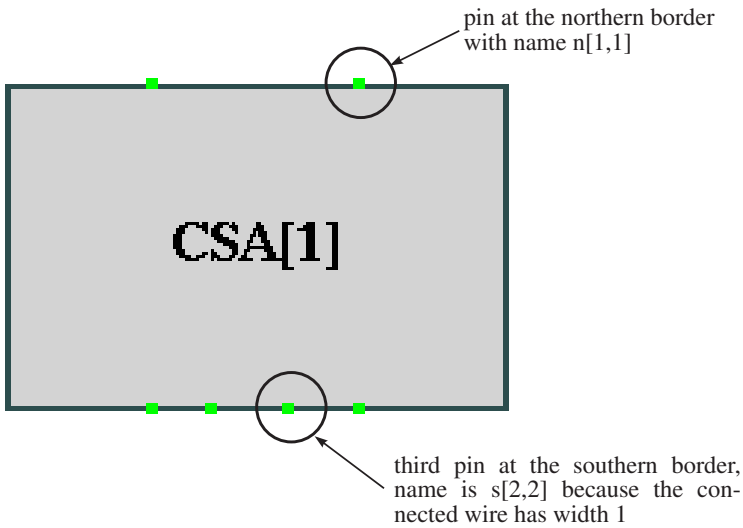

Figure 12.8: Graphical representation of a discrete macro cell in the case of a 1 bit conditional sum adder

The graphical representation of a discrete macro cell will be derived from the corresponding schematic. When the schematic is saved an entry in the discrete macro list will be generated. The position of the pins is given by the relative positions of the connections of the corresponding wires with the border of the schematic. The pin names are generated automatically. They consist of the side of the schematic to which the pin belongs and an interval which describes the position and the width of the pin. The width of the pin is given by the difference between the upper and lower bound of the interval plus one. The order of the intervals is from left to right at the northern and the southern border and from top to bottom at the western and eastern border of the macro. The order of the intervals is important in the case of rotations of the macro. Examples for pin names are the following:

⇝ 12.2.2

`n[0,0]` first pin at the northern border of a macro, the connected wire has the width 1

`e[3,4]` this pin represents the connection of the fourth and fifth wire at the eastern border, which are bundled in a single wire of width 2

`s[1,1]` second pin at the southern border

`w[0,2]` first pin connector at the western border of the macro, representing a bundle of width 3

In the graphical representation of a macro cell, the pin names are not displayed, only the pin connectors are shown. Macro cells normally have no power supply pins, when they are selected from the macro library. Power supply pins are generated by invoking an appropriate tool (c.f. chapter 8).

The third group of cells you can use are parameterized macros, which depend on the free parameters of the schematic. For these macros in general there exists no definition from a basic library or a schematic input. That is why its size and the position of its pins are unknown when it is selected. Therefore it is displayed as a rectangle of default size which has no pin connectors first.

⇝ 12.4.1 Pins are created when you connect wires to the border of the macro which is treated like the border of the schematic. The size of a parameterized macro

⇝ 12.3.3 can be changed after it is placed within the workarea in order to create a more intuitive representation of the schematic.

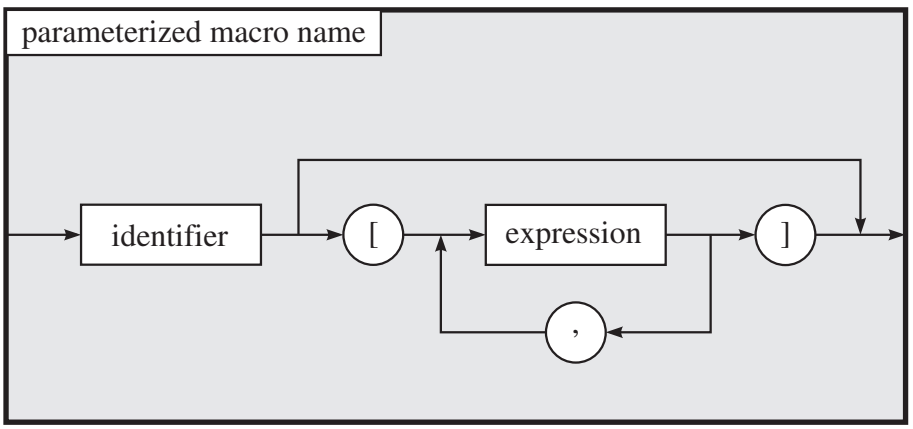

Figure 12.9: Syntax of parameterized macro names

To select a new parameterized macro, move the pointer into the third cell list window near to the entry `--- New Cell ---` and press the left mouse button if it is highlighted. The input window will popup and you are prompted to enter a new name for a parameterized macro cell:

```
Please enter Parameterized Name:  <          >
```

Now you can type in the name of a new parameterized macro cell. As shown above for the names of schematics there are syntactical rules which must be considered when selecting the name. The diagram in figure 12.9 shows the basic structure of a parameterized name.

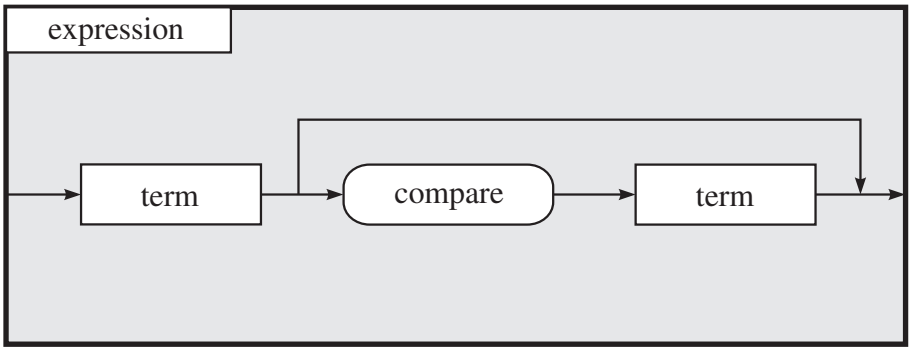

Figure 12.10: An expression consists of a single term or the comparison between two terms

The diagram in figure 12.10 defines that the name of a parameterized macro consists of an identifier followed by an optional list of parameters, separated by commas. Each parameter is given by a construct named expression which will be refined in the following. The diagram implies that a macro may have an infinite number of parameters, but as for schematic names, there is the limitation to a maximum of 32 parameters.

Each parameter of the type expression is defined by the following diagrams. We use the hierarchical description known from programming languages in order to describe the priorities between the available operators. In a first step an expression is refined according to the diagram in figure 12.10, where we use a construct named term. An expression may be a single term, or it may be the comparison of two terms. As shown in section 12.1 a comparison

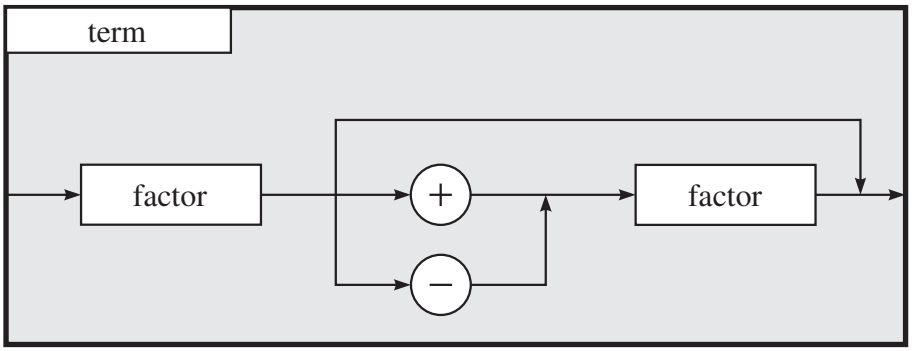

Figure 12.11: Syntactical structure of term which is the additive combination of two factors

operator may be one of `< <= >= > == !=`. The comparison between two terms is evaluated to zero or one, if it is false or true.

The parts of an expression are described by the construct term, for which the syntax diagram is given in figure 12.11. A term represents a single factor or the combination of two factors with the use of an additive operator (`+` or `-`). We have chosen this representation to express the higher priority of the elements in factor with regard to addition and subtraction.

A factor is defined by the syntax diagram in figure 12.12. It combines two objects of the type unit by using a multiplicative operator like multiplication (`*`), division (`/`), power (`^`) and division modulo (`%`). The use of a new object of type unit introduces a new priority level for this construct.

The object with the highest priority is called unit. Its syntactical structure is given by the diagram in figure 12.13. Beside the call of a standard function as they are listed in section 12.1 a unit can be an identifier or an unsigned integer. In the diagram any identifier is allowed but within a schematic you may only use the identifiers which appear as parameters in the schematic name.

⇝ 12.1

Legal names for parameterized macros are the following:

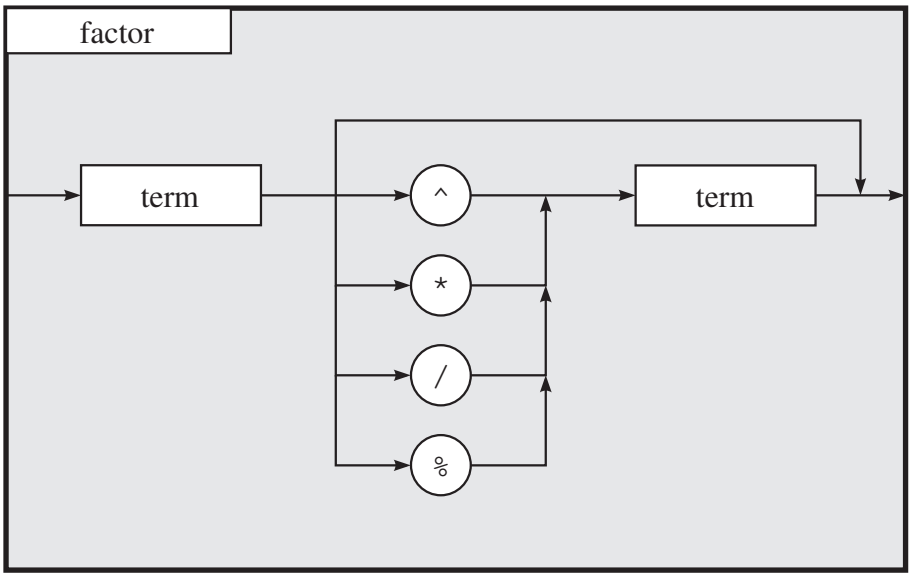

Figure 12.12: Syntactical structure of factor which is the multiplicative combination of two units

```
CSA[n/2]
SEL[upper(k/2)]
GChannel[n,min(n-2^i,2^i),i<(log(n)-1)]
```

After the selection of a cell from one of the three cell list windows, the corresponding graphical representation of the cell type is shown within the workarea. The cell follows the movement of the pointer in order to choose the position where you want to place it. During the motion you may notice that the cell changes its colour from red to grey. This indicates that you have selected an illegal position for the placement.

Illegal positions are overlappings of the current cell with other cells, with the border of the schematic or with wires. If the cell is on an illegal position you can not place it by pressing the left mouse button. This will have no effect and the cell will still be movable with the pointer. If you no possibility to place the cell you can abort placement mode by pressing the right mouse button within the workarea.

If you have selected a legal position for the cell and dropped it or if you aborted the cell placement mode, the three cell list windows are popped up

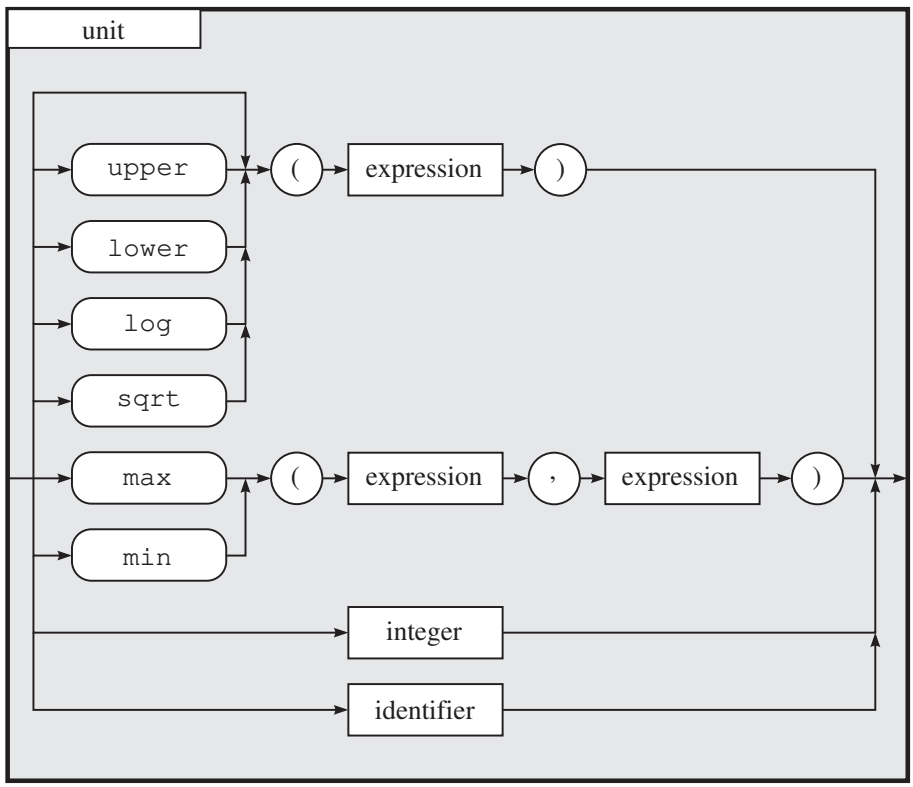

Figure 12.13: Syntactical structure of an object of type unit

again in order to select another cell. You can abort this mode by pressing the right mouse button within one of the three list windows. This operation terminates the enter cell mode and you return to menu selection mode.

12.3.2 Move and Rotate Cells

With the function `Move` from the `-Cells-` submenu you can select placed cells and macros within the workarea and move them to a new position or change their orientation. If you select this entry from the menu and move the pointer into the workarea you are in cell selection mode. This is indicated by the nearest cell to the pointer being highlighted in blue. If you press the left mouse button this cell will be selected.

Note: After the selection of a cell for movement or rotation, it is isolated from its context within the schematic. The wires are no longer connected to the cell. We recommend that you delete all wires, which are connected

⇝ 12.4.2

to the cell before you move it to a new position. But in most cases you will select cells from the cell list windows, place them within the schematic and change their positions or orientations before you enter the signal wires.

After the selection the instance is in cell placement mode and it can be moved freely within the workarea as we have shown it above in section 12.3.1 when we entered a new cell. The movable cell changes its colour, if you select an illegal position, where it intersects another cell, the border of the schematic or a wire. It is impossible to place it at an illegal position, the left mouse button has no effect in this case. If the cell is at a legal position, you can fix it there by pressing the left mouse button. You also can abort placement mode by pressing the right mouse button. In this the case the movable cell is returned to its old position and cell selection mode is activated again in order to select another cell for movement operation.

⇝ 12.3.1

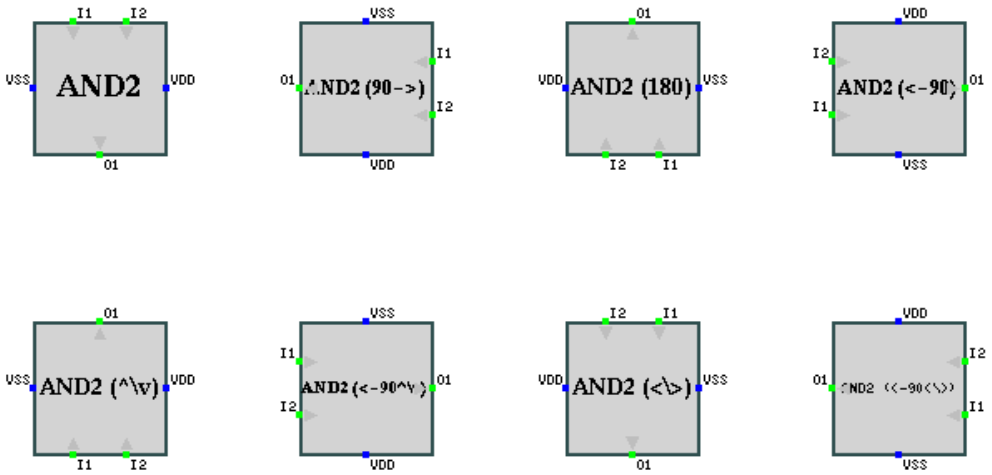

Figure 12.14: Sequence of possible orientations of a basic cell

With the help of this menu entry you can not only select cells and move them to a new position, you also can change their orientation. After the selection of a cell it is in placement mode. During the movement you can select a new orientation for it by pressing the middle mouse button several times. Each button press event changes the orientation of the cell to one of the eight possible representations shown in figure 12.14.

The current orientation is indicated by a suffix in the cell name. The description of these suffixes is given in figure 12.14. It is not a part of the cell name. You can notice this, when you rename a cell, which is not in normal orientation. In this case the cell with the new name has the same orientation as before and keeps its suffix.

⇝ 12.3.5

If you have chosen the desired orientation, you can move the cell to a new location and fix it there by pressing the left mouse button. If you do not want to change the orientation of the selected cell, you can abort the operation by pressing the right mouse button. In this case the cell is moved to its original position and orientation. You are in cell selection mode and you can choose another cell for movement or rotation.

You can terminate this operation by pressing the right mouse button within the workarea, if you are in cell selection mode. If you are in cell placement mode, i.e. a cell is following the motion of the pointer, you must abort this first (right mouse button).

12.3.3 Resize Cells

With the function `Resize` from the `-Cells-` submenu you can select placed cells or macros within the schematic and change their sizes. In most cases this function is used to resize parameterized macros, for which the system selects the default size. Changing the size of these macros is useful for creating an intuitive representation of a schematic. You can select appropriate sizes for the macros in order to indicate their different priorities.

If you activate the resize operation in the menu, you are in cell selection mode. Move the pointer into the workarea and choose the cell which you want to resize (nearest cell to the pointer is highlighted in blue) by pressing the left mouse button.

Now you are in resize mode which is indicated by a rubberbanding frame following the motion of the pointer. This frame is connected at the upper left corner of the selected cell. Its opposite corner is at the position of the pointer indicating the new size of the cell. You can confirm the current size

by pressing the left mouse button or you can abort the resize mode with the right mouse button.

If you have selected a new size for the macro, it is displayed now according to your selection. The resized macro is now in the cell placement mode and you have to choose a new position for it. This is necessary because the resizing of the macro could lead to overlappings with other cells, wires or the schematic border. Move the cell to a legal position (illegal positions are indicated by a change of the colour of the cell from red to grey) and place it there by pressing the left mouse button. If you abort cell placement mode, the cell is returned to its old position and its original size is restored.

After that you are in the cell selection mode again and can choose another cell for resize operation or you can terminate the resize operation by pressing the right mouse button within the workarea.

Note: As in the case of moving and rotating cell we recommend that you delete all wires, which are connected to the cell, before you select it for resizing.

12.3.4 Copy Cells

With the function `Copy` from the `-Cells-` submenu you can create a copy of an already placed cell or macro. This function is very useful, if you have changed the attributes of a cell (size or orientation) and you need a new cell with the same attributes. Without this function you would have to enter a new macro from the cell list windows, place it within the schematic and change the attributes.

⇝ 12.3.2

This function is very similar to the move operation from section 12.3.2. If you have activated the copy operation, you are in cell selection mode and can choose the macro, from which you want to create a copy.

After the selection with the left mouse button, a second instance of this cell is created. Then you are in the cell placement mode and can move the new macro to the desired position (confirm it by pressing the left mouse button). You can abort cell placement mode with the right mouse button. In this

case the newly created cell will disappear.

After that you are in cell selection mode again and you can select another cell to be copied. If you do not want to copy more macros, you can abort this function by pressing the right mouse button within the workarea.

12.3.5 Rename Cells

With the function `Rename` from the `-Cells-` submenu you can change the names of macros. This function is useful, if you have made a copy of an instance and you want to change its parameters. The size and the orientation are not modified by this operation.

If you activate this function, you are in cell selection mode in order to choose the cell, for which you want to change the name. Move the pointer near to the desired macro and press the left mouse button to select it.

Note: this function is only applicable with macros, not with basic cells. If the currently highlighted instance is a basic cell, pressing the left mouse button has no effect. You will remain in cell selection mode. You can not simply exchange the names of basic cells, because they have a fixed interface which is given by the descriptions in the selected library. This information is global to all users of this library.

After the selection of a macro cell, the input window will popup and ask you for the new name of the macro:

```
Please enter New Instance Name:<        >
```

Type in the new macro name and confirm your input by pressing the return key. Note that you have to consider the syntactical rules for macro names as they are described in section 12.1 and 12.3.1. If you typed in a syntactically wrong name, the name of the selected macro will not be changed and an error message is displayed in the message window:

⇝ 12.1,12.3.1

```
!  Error !  "SHUFFLE[n" is illegal Instance-Name
```

In this case the input window will popup again, and you can correct your

input. If you do not want to change the name of the selected macro any longer, you can cancel the input window by pressing the return key to an empty input line.

If your input is correct, the name of the selected macro is changed in this way. In both cases (cancelling the input window or choosing a correct name) the input window is popped down and you return to cell selection mode in order to select another macro, for which you want to change the name. You can abort this mode and thereby abort the whole rename function by pressing the right mouse button within the workarea.

12.3.6 Delete Cells

With the function `Delete` from the `-Cells-` submenu you can erase cells and macros from the current schematic. After activating this function you are in cell selection mode. Move the pointer into the workarea near to the cell you want to delete and press the left mouse button, when it is highlighted.

Note: only the cell or macro is deleted. The connected wires remain in the schematic. If you do not need them, call the function `Delete` from the `-Wires-` submenu .

⇝ 12.4.2

After the deletion of a cell you are in cell selection mode in order to delete more cells. You can abort this mode and the delete function by pressing the right mouse button within the workarea.

Note: the function is automatically terminated, if you have deleted the last cell in the schematic. Therefore you can not activate this function, if there is no cell or macro in your schematic.

12.4 Wires

12.4.1 Enter Wires

With the function `Enter` from the `-Wires-` submenu you can draw wires in order to connect cells and macros. Wires are build up of sequences of horizontal and vertical edges. The graphical representation of a wire may represent a bundle of single parallel wires. The number of these single wires is described by the width of the graphical wire. Concerning the width the wires can be clssified into three groups:

- wires with constant width
- wires with arithmetical expressions over the schematic parameters
- wires with formal variables

For the use of these wire types we define the following rules:

- at the pin connectors of basic cells and discrete macros you may only connect wires with a constant width. In the case of basic cells the width is always 1.
- at the borders of parameterized macro cells and the schematic you may connect wires with a width of all three types.

If the width of the wire is not equal to 1, it is displayed beside a small slash at the middle of the wire. The following figure 12.15 shows wires of all three types.

The enter wire function has to perform the following tasks

- selection of the start and end point of the wire
- determination of the wire width
- interactive error handling

If you activate the enter wire function, you first have to select the start point for a wire. If you move the pointer into the workarea you notice a crosshair cursor following the pointer motion. Because of the high resolution of the

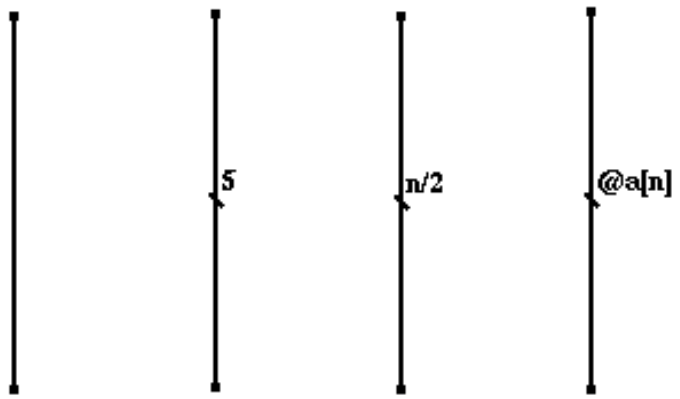

Figure 12.15: Wires with different widths

screen it is difficult for the user to select exactly a certain pixel position. Therefore the crosshair cursor not only specifies a pixel position, but it covers a region of pixels. If you move it near to a special position within the schematic, it will snap to that position. You can notice this behaviour, if you move the pointer slowly near to one of the following special points

- point on a wire (but not its start or end point)
- start or end point of a wire
- pin connector
- point on schematic or parameterized macro border

If the cursor is near to any of these points, the system will choose the coordinates of it. If this is not the case the position is the center of the cursor. The special points have increasing priority in the given order, i.e. a pin connector has higher priority than a point on a wire.

You can fix the start point of the wire by pressing the left mouse button. After that you can move the pointer to the end point of the wire. During the motion of the pointer you will notice a rubberbanding line connecting the start point with the current pointer position. The shape of this line indicates the wires which will be inserted, if you fix the end point. In some cases this

line will disappear. This indicates that you have chosen a connection of two points, which is impossible by insertion of at most two wires (e.g. one wire would intersect a cell). Note that the line will also disappear, if you connect two wires with different widths.

If you have chosen a legal connection (the rubberbanding line is visible), you can fix the end point by pressing the left mouse button again. If you do this for an illegal connection, the selected end point will become the start point for a new connection, i.e. the rubberbanding line will now be drawn from the expected end point to the current pointer position. You also can abort the selection of the end point by pressing the right mouse button and return to the selection of a new start point.

If you have drawn a legal connection between two wire points, the rubberbanding line will disappear and the system will now determine the width of this wire. In some cases this can be automatically derived from the points you have connected:

- if one of the wire points is a pin connector of a basic cell, the wire gets the width 1.
- if one of the wire points is a pin connector of a discrete macro cell, the wire gets the width of this pin.
- if one of the points is located on a wire, the new wire gets the same width as this wire.

In the other cases the system can not automatically determine the width of the new wire. Then the input window is popped up and you have to specify the wire width to the following prompt:

```
Please enter Wire Width:<        >
```

For the selection of the wire width you have to consider the following syntactical rules. As mentioned earlier the width of a wire may depend on the free parameters of the schematic. It may be an arithmetical expression in these parameters or it may be described by a wire variable. The diagram in figure 12.16 shows the syntactical structure of a wire width.

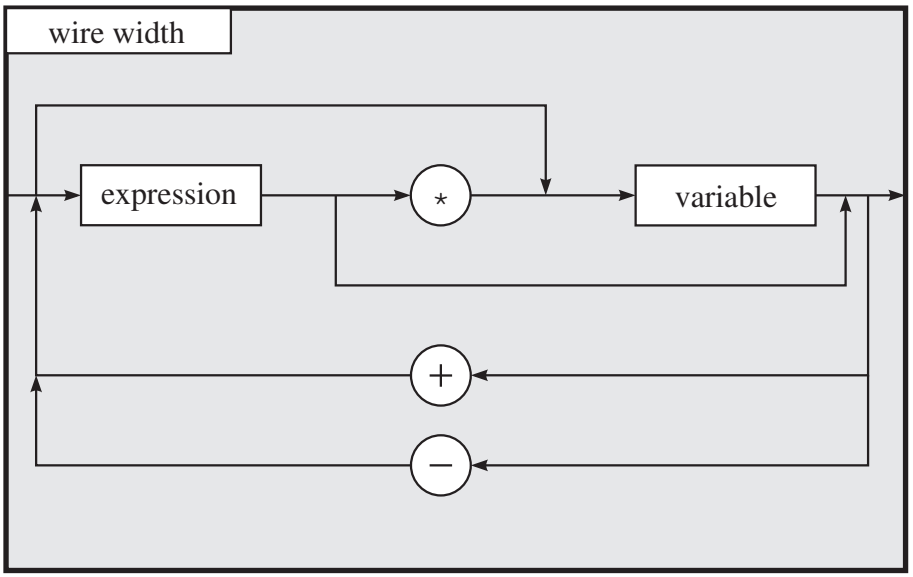

Figure 12.16: Syntactical structure for the width of a wire

This diagram shows that the width of a wire may be a simple expression as it is given by the diagrams in figure 12.10 to figure 12.13. But it also may be a linear combination of variables which have expressions as coefficients. The restriction to a linear combination is necessary, because we compute the values of the wire variables by solving an equation system.

The syntactical structure of a wire variable is given by the diagram in figure 12.17. A variable is introduced by the special character `@` followed by a construct which is similar to a parameterized macro name. Especially a wire variable may depend on parameters which are given by expressions over the schematic parameters. This allows you to specify flexible variables within a recursive description, whereas a variable without parameters would have the same value for all stages of the recursion.

The following inputs are legal wire widths:

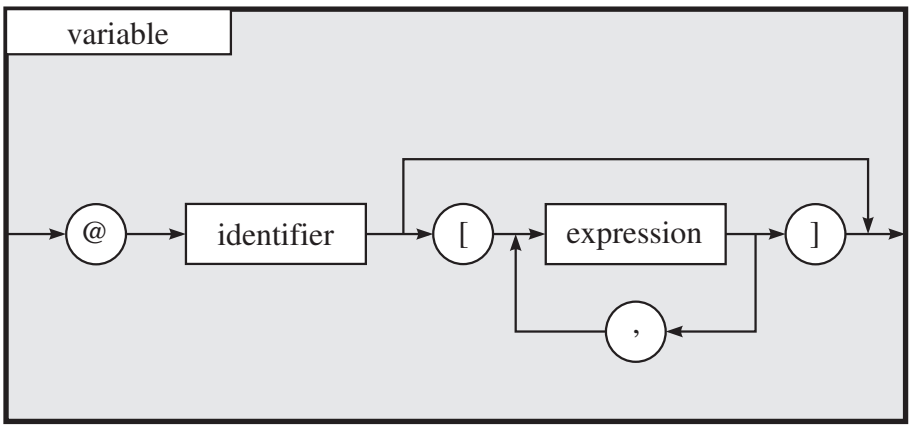

Figure 12.17: Structure of a wire variable

`5`	constant value
`n/2+1`	arithmetical expression
`@t`	constant wire variable
`@s[n]`	variable depending on schematic parameter
`@w[n>0,m==1]`	variable depending on comparisons
`@s[n]+n*@t[n-1]`	linear combination of variables

The following inputs are illegal wire widths:

`-1`	width may not be negative
`@s[@t[n]]`	illegal parameter
`@s[n/2]*@t[n]`	not a linear combination of variables
`0.5*@t[n-1]`	coefficients must be integers

If you enter a correct wire width and confirm this by pressing the return key, the new connection is drawn as one or two yellow edges. If the width is not equal to one, it is drawn as a label in the middle of each edge which is longer than 50 pixels (for very small edges the width is not shown on the screen, but you can look at it, if you zoom in the schematic with the help of the functions in section 12.7.1).

⇝ 12.7.1

After the new wire is created, you can draw another connection. Here the end point of the previous wire is taken as the start point for the next wire. This enables you to draw a sequence of more than one wire segment. If you do not want to use the end point of the previous wire as the start point

for the next wire, press the right mouse button once. Then the crosshair at the end point and the rubberbanding line will disappear and you can select a new start point for the next wire. Especially in some cases it is impossible to use the end point of the previous wire as the new start point (e.g. end point is located on the schematic border), then you will see no rubberbanding line (indicating that there is no legal connection possible). Then the next point selected will be the new start point of the next wire.

If you have entered an illegal wire width, the input window will popup again and you can correct your input. If you press the return key to the empty input window, the just created connection will be deleted.

You can terminate the enter wire function by pressing the right mouse button within the workarea. Note that you have to press it twice, if you are selecting the end point of a wire (rubberbanding line will follow the pointer motion).

12.4.2 Delete Wires

With the function `Delete` from the `-Wires-` submenu you can erase wires from the current schematic. After activating this function you are in the wire selection mode. If you move the pointer into the workarea, the wire nearest to the pointer position is highlighted (it changes its colour from yellow to cyan). Now you can delete this wire by pressing the left mouse button.

After the deletion of the wire you remain in wire selection mode in order to delete more wires. This mode and the delete wire function can be aborted by pressing the right mouse button within the workarea.

Note: the function is automatically terminated, when you have deleted the last wire in the schematic. That is why you can not activate this function, if there is no wire at all in your schematic.

12.5 Comments

12.5.1 Enter Comments

With the function `Enter` from the `-Comments-` submenu you can insert comment text into your schematic. If you activate this function, the input window will popup with the following prompt:

```
Please enter Comment Text:<          >
```

Now you can type a comment text which may contain every printable character. The maximum length of the text may not exceed 80 characters. If you have to enter more text, you must split it in several comment lines.

If you confirm your input by pressing the return key, the text will appear within the workarea. It is now movable with the pointer and can be placed at any position in your schematic (inside cells or macros, across wires etc.). Move the text to the desired position and drop it there by pressing the left mouse button. You can abort the placement operation by pressing the right mouse button (the comment text is deleted in this case).

After that the input window will popup again and you can type in another comment text. If you do not want to place more comment texts you can abort this function by pressing the return key to the empty input window.

12.5.2 Delete Comments

With the function `Delete` from the `-Comments-` submenu you can erase placed comment text from your schematic. If you activate this function you are in comment selection mode. If you move the pointer into the workarea, the comment text nearest to the pointer position is highlighted (its colour changes from white to cyan).

Now you can delete this comment by pressing the left mouse button. After the deletion of a comment you return to comment selection mode in order to delete more comments. You can abort this mode and the delete function by pressing the right mouse button within the workarea.

Note: the function is automatically terminated, if you have deleted the last comment in the schematic. Therefore you can not activate this function, if there is no comment at all in your schematic.

12.6 Equations

12.6.1 Enter Equations

With the function `Enter` from the `-Equations-` submenu you can insert additional equations about variables which are used for the description of
⇝ 12.4.1 the width of bundles of wires . In some cases the equations automatically derived by the system are not sufficient to compute a unique solution. In section 5 we describe, how the system generates equations about the wire variables.

Entering an equation is very similar to entering a comment text. If you activate this function, the input window is popped up with the following prompt:

```
Please enter New Equation:<       >
```

and you can type in an equation, for example

```
@a[n] = @b[n]
@s[n]+@t[n] = 2*@t[n-1]
@t[n] = 10
```

The syntactical structure of both sides of the equation is given by the dia-
⇝ 12.4.1 gram in figure 12.16 .

If your input is correct, the equation will appear in the workarea and is movable with the pointer. Just as comment text you can place it anywhere in your schematic and fix its position by pressing the left mouse button. Note that equations are drawn in yellow, whereas comment text is displayed in white.

If you typed in a syntactically incorrect equation the system will respond with an appropriate error message in the message window:

```
!!!Error!!!  Equation ...  is syntactically incorrect
```

Then the input window will popup again in order to correct your input. If you do not want to enter another equation, you can abort the function by pressing the return key to the empty input window.

12.6.2 Delete Equations

With the function `Delete` from the `-Equations-` submenu you can erase placed equations from your schematic. If you activate the function, you are in equation selection mode. If you move the pointer into the workarea, the equation nearest to the pointer is highlighted (it changes its colour from yellow to cyan). Note that you can only select equations, i.e. comment text will not be highlighted.

The highlighted equation can now be deleted by pressing the left mouse button. After that you remain in equation selection mode in order to delete more equations. You can abort this mode and the delete equation function by pressing the right mouse button within the workarea.

Note: the function is automatically terminated, when you have deleted the last equation in your schematic. That is why you can not activate it, if there is no equation at all in your schematic.

12.7 Views

12.7.1 Zooming

With the functions `Zoom In` and `Zoom Out` from the `-Views-` submenu you can select a viewport and change the graphical representation of your schematic.

If you activate the function `Zoom In` you are in viewport selection mode. Now move the pointer into the workarea and select a start point for a rectangular viewport. This need not be the upper left corner of the viewport, according to the selection of the second point it will be interpreted appropri-

ately. After fixing the start point by pressing the left mouse button, move the pointer to the position of the opposite corner of the desired viewport. You notice a rubberbanding frame following the motion of the pointer. This frame indicates the size of the selected viewport and implies the factors by which the schematic will be scaled.

If you confirm the selection of the opposite corner by pressing the left mouse button, the schematic will be scaled and redrawn. The upper left corner of your selected viewport will be moved into the upper left corner of the workarea. You remain in viewport selection mode for further zooming operations. Each zooming operation is pushed onto a zooming stack, so that we can return to any previously selected viewport. The currently active viewport will be shown in the control window as a small yellow rectangle, representing the position and the size of the viewport relative to the size of the whole schematic.

The selection of the corners of a viewport can be aborted by pressing the right mouse button. If you do this during the selection of the first corner, the zooming function is terminated and you return to menu selection mode. If you press the right mouse button during the selection of the opposite corner, the start point is cancelled and you can select a new first point of the viewport.

The inverse function `Zoom Out` can be used to restore the viewport on the top of the zooming stack. This function has no effect, if the zooming stack is empty. If you want to restore the original size of the schematic, you can directly call the function `Normal` which clears the zooming stack in one step.

12.7.2 Scaling

With the function `+10%`, `-10%`, `+50%` and `-50%` from the `-Views-` submenu you can change the size of the schematic without moving its upper left corner to a new position. The scaling factors are 1.1, 0.9, 1.5 and 0.5 respectively.

After calling these functions for several times you can return to the original size of the schematic with the help of the function `Normal`.

12.8 Miscellaneous

With the function `All Objs` from the `-Views-` submenu you get a representation of your schematic which will intuitively show the relation to the underlying mathematical calculus. The graphical represenation resembles an equation, where the left side is the macro representation of the current schematic and the right side is the schematic itself. This denotes that the macro is refined (refinement operator is indicated by an arrow) by the drawing of the schematic.

We do not recommend to use this representation during the input of the schematic, because the frame is scaled down, so that there is not enough space for comfortable input operations. You can return from this representation to the normal mode, if you call the functions `Normal` and `Home`. The call of `Normal` will restore the original size of the schematic and `Home` will move its upper left corner into the upper left corner of the workarea.

The function `Redraw` can be used to refresh the drawing of the schematic within the workarea. This function is useful, if there remains some dirt from entering and deleting objects as macros, wires or comments. Some functions automatically do a redraw operation, so that you do not have to call it very often.

The function `Inverse` changes the background colour of the workarea from black to white. It also changes the colours of macros, wires, comments and equations appropriately. This is useful, if you want to make screendumps from the graphical editor as we did it during the design example in section 4.8.

To terminate the editor session you have to terminate any currently active editor function first. In most cases this can be done by pressing the right mouse button at most twice within the workarea (see also the description of the appropriate function). After that you are in menu selection mode and you can finish your editor session by pressing the right mouse button within the menuline. Now the editor menu will be closed and you return to the main menu of the CADIC system. From there you can select another

tool or you can terminate the whole system call by selecting the entry `Exit` in the main menu.

Bibliography

[Bar78] M. Barbacci et.al. The Symbolic Manipulation of Computer Descriptions. Technical report, Dept. of Computer Science, Carnegie–Mellon University, 1978.

[Biw94] M. Biwersi. μsic – ein kleiner Silicon-Compiler. Master's thesis, Fachbereich Informatik, Universität des Saarlandes, Postfach 15 11 50, 66041 Saarbrücken, FRG, 1994.

[Bur94] Th. Burch. *Eine graphische Arbeitsumgebung für den parametrisierten Entwurf integrierter Schaltkreise.* PhD thesis, Fachbereich Informatik, Universität des Saarlandes, 1994.

[CAD92] CADENCE. *Cadence Design System Manual*, 1992.

[CF86] E. Clarke and Y. Feng. ESCHER – A Geometrical Layout System for Recursively Defined Circuits. In *Proceedings of the 23rd Design Automation Conference*, pages 649–653, June 1986.

[CNR87] H.A. Choi, K. Nakajima, and C.S. Rim. Complexity Results for vertex–deletion Graph Bipartization and Via Minimization Problems. In *Proceedings of the 25th Annual Allerton Conference on Computing, Communication and Control*, September 1987.

[Com87] EDIF Steering Committee. *EDIF Electronic Design Interchange Format Version 2 0 0.* Electronic Industries Association, Washington D.C., May 1987.

[DBR+88] P. J. Drongowski, J. R. Bammi, R. Ramaswamy, S. Iyengar, and T. H. Wang. A Graphical Hardware Design Language. In *Proceedings of the 25th Design Automation Conference*, pages 108–115, 1988.

[Fet95] Th. Fettig. Lokale Plazierung in CADIC. Master's thesis, Fachbereich Informatik, Universität des Saarlandes, 1995. to appear.

[Gen90] GenRad. *HILO Reference Manual*, 1990.

[GL85] S. M. German and K. J. Lieberherr. Zeus: A Language for Expressing Algorithms in Hardware. *IEEE Transactions on Computer Aided Design*, pages 55–65, Februar 1985.

[Gra92] B. Grande. Verfahren zur hierarchischen Schichtzuweisung und ihre Implementierung in CADIC. Master's thesis, Fachbereich Informatik, Universität des Saarlandes, 1992.

[HMZ91] G. Hotz, P. Molitor, and W. Zimmer. On the Construction of Very Large Integer Multipliers. In *Proceedings of EURO ASIC 91*, pages 266–269, May 1991.

[Hot65] G. Hotz. Eine Algebraisierung des Syntheseproblems für Schaltkreise. *EIK Journal of Information Processing and Cybernetics*, 1:185–205,209–231, 1965.

[Hot74] G. Hotz. *Schaltkreistheorie.* de Gruyter Lehrbuch. Walter de Gruyter Verlag, 1974.

[KCS88] Y.S. Kuo, T.C. Chern, and W. Shih. Fast Algorithm for Optimal Layer Assignment. In *Proceedings of the 25th Design Automation Conference (DAC88)*, pages 554–559, June 1988.

[KMO89] R. Kolla, P. Molitor, and H. G. Osthof. *Einführung in den VLSI–Entwurf.* Leitfäden und Monographien der Informatik. B.G. Teubner Verlag, Stuttgart, 1989.

[Knu73] D.E. Knuth. *Sorting and Searching, The Art of Computer Programming.* Addison–Wesley, 1973.

[Kol86] R. Kolla. *Spezifikation und Expansion logisch topologischer Netze*. PhD thesis, Fachbereich Informatik, Universität des Saarlandes, 1986.

[Lim82] W. Y.-P. Lim. HISDL – A Structure Description Language. *Comm. ACM*, Vol. 25:823–830, November 1982.

[LSU89] R. Lipsett, C. Schaefer, and C. Ussery. *VHDL: Hardware Description and Design*. Kluwer Academic Publishers, 1989. 300 Seiten.

[LV83] W.K. Luk and J. Vuillemin. Recursive Implementation of Optimal Time VLSI Integer Multipliers. In *Proceedings of IFIP Congress 83*, pages 155–168, 1983.

[MN89] K. Mehlhorn and S. Näher. LEDA a Library of Efficient Data Types and Algorithms. Technical Report TR–A 04/1989, FB 10, Universität des Saarlandes, 1989.

[Mol86] P. Molitor. *Über die Bikategorie der logisch topologischen Netze und ihre Semantik*. PhD thesis, Fachbereich Informatik, Universität des Saarlandes, Postfach 15 11 50, 66041 Saarbrücken, FRG, 1986.

[Mol87] P. Molitor. On the Contact Minimization Problem. In *Proceedings of the 4th Annual Symposium on Theoretical Aspects of Computer Science (STACS87)*, pages 420–431, February 1987.

[Mol93] P. Molitor. A Hierarchy Preserving Hierarchical Bottom-up 2-Layer Wiring Algorithm with respect to Via Minimization. *INTEGRATION, the VLSI Journal*, 15:73–95, 1993.

[Sch92] J. Schnabel. Generierung der Stromversorgung in CADIC. Master's thesis, Fachbereich Informatik, Universität des Saarlandes, 1992.

[Sch96] Ch. Scholl. *Logiksynthese unter Ausnutzung funktionaler Eigenschaften*. PhD thesis, Fachbereich Informatik, Universität des

Saarlandes, Postfach 15 11 50, 66041 Saarbrücken, FRG, 1996. to appear.

[Sha82] M. Shadad et.al. VHSIC Hardware Description Language. *IEEE Transactions on Computer Aided Design*, Vol. 18, Februar 1982.

[Skl60] J. Sklansky. Conditional-sum addition logic. *IRE-EC*, 9:226–231, 1960.

[SSC82] J. M. Siskind, J. R. Southard, and K. W Crouch. Generating Custom High Performance VLSI Designs from Succinct Algorithmic Descriptions. In *Proc. Conf. on Advanced Research in VLSI*, pages 28–40, Januar 1982.

[SYS90] TANGENT SYSTEMS. *Tangate Reference Manual*, 1990.

[Wal64] C.S. Wallace. A Suggestion for a Fast Multiplier. *IEEE Transactions on Computers*, 13:14–17, 1964.

[Wan95] G. Wannemacher. Generierung eines symbolischen Layouts in CADIC. Master's thesis, Fachbereich Informatik, Universität des Saarlandes, Postfach 15 11 50, 66041 Saarbrücken, FRG, 1995. to appear.

[Wir82] N. Wirth. Hades: A Notation for the Description of Hardware. Technical report, ETH Zentrum Zurich, August 1982.

Index

!=, 269
(180), 86
(90->), 86
(<\>), 86
(<-90), 86
(<-90<\>), 86
(<-90^\v), 86
(^\v), 86
*, 269
+, 269
-, 269
/, 269
<=, 269
<, 269
==, 269
>=, 269
CELLDIR, 20
DAGDIR, 20, 257
EDIFDIR, 21
HILODIR, 21
MACROLIB, 21
NBSDIR, 21
PARLIB, 21
PWRDIR, 21
SIMDIR, 21, 157, 163
SLICEDIR, 21
Schematic Editor
    exit, 107
TECHDIR, 20
VENUSDIR, 21
^, 269

basic cell, 263
basic topographical net, 202
bicategorial expression, 30
bifunctor, 31
BTG, 202

cell
    copy, 52, 64, 274
    delete, 276
    enter, 263
      basic, 36, 72, 263
      discrete macro, 264
      parameterized, 47, 49, 61, 265
    flip, 85, 271
    move, 271
    orientation, 272

placement mode, 52, 53, 63, 65
rename, 79, 275
resize, 51, 63, 273
rotate, 85, 271
selection mode, 53, 64
Cgraph, 112
edge information, 148
node information, 145
node types, 147
scan, 146
channel, 202
comment
delete, 98, 283
enter, 70, 283
conditional sum adder, 58
control area, 23

DAG, 5, 109
best matching, 122
building, 113
hierarchy check, 136
hierarchy error, 137
logarithmic size, 122
structure check, 136
trace down, 141
trace up, 143
tracing path, 143
traversal, 112
treenode, 110
visualization, 133
data size, 116
design environment, 6
design tools, 6
directed acyclic graph, 109
discrete macros, 264
divider, 232

EDIF, 25
environment area, 23
equation
delete, 285
enter, 105, 284

functional behaviour, 155

geometrical layout, 201

hardware description language (see also HDL), 25
HDL, 25
EDIF, 25
VHDL, 25
Zeus, 25
hierarchical transition, 111

instance, 110
integrated tools, 6

layer assignment, 173
bus via, 176
dual graph, 175
dual mode, 180
multi layer wires, 183
problems, 174
refining wires, 183
remove vias, 186
via minimization, 175
wiring, 176
layers, 173
layout, 201

area protocol, 205, 209
graphical expansion, 206, 211
PostScript output, 215
tracing, 212
views, 213
logic simulation, 155
logic topographical net, 28
horizontal composition, 28
vertical composition, 28
logic topological net, 29

main menu, 23
merging circuit, 219
message area, 23
mode
cell selection, 51, 62
modification time, 115
multiplier, 227

navigation, 6, 144
net equation, 30
net variable, 30
best matching, 122

odd even mergesort, 217

parameter
setting value, 118
parameter syntax, 268
parameterized macros, 265
pattern file, 168
place&route, 201
power supply, 190
sizing, 192
topology, 190
PROTDIR, 209

redundant representation, 233
river routing, 202

schematic
clear, 262
delete, 263
list, 257
name, 259
new, 35, 49, 59, 257
open, 96, 257
redefinition, 260
save, 57, 71, 261
save as, 97, 262
Schematic Editor
start, 34
shuffle circuit, 91, 221
simulation
display mode, 166
next pattern, 171
pattern file, 168
pattern list, 168
prepare design, 157
single pattern, 159, 164
sorting circuit, 217
specification
parameterized, 3
recursive, 4
specification methods, 25
standard functions
`log`, 256
`lower`, 256
`max`, 256

`min`, 256
`sqrt`, 256
`upper`, 256

treenode, 110

VHDL, 25
views, 110
scaling, 152, 286
zoom in, 150, 285
zoom out, 151, 286
visualization, 6

Wallace tree multiplier, 26
wire
delete, 98, 282
enter, 54, 66, 277
parameterized, 55
pin connection, 39, 43, 74
projection, 70
variable, 68
equation, 105
width, 54, 68, 277
wire variables, 124
advantages, 132
basic equations, 127
deriving equations, 125
display equations, 129
display variables, 131
equation system, 124
generic descriptions, 132
illegal solution, 128
reusable structures, 133
work area, 22
X–Window System, 17