9

Geometrical Layout Design

9.1 Introduction

At the moment, the circuit layout in CADIC is only a topological design which is at the gate level and thereby independent of concrete technologies. In the fact, this layout design handles the problem of finding a "good" topographical representative of a logic topological net. The construction of the layout bases on the circuit representation by bicategorial expressions as it is shown in chapter 8. Here we also use the function library for the slicing of the Cgraphs in order to build the data structure shown in figure 8.1. If this structure has been calculated earlier and the Cgraphs have been changed by another synthesis tool as for example the power supply or the layer assignment tool, the slicing is newly created. The the generation of the layout is performed in the following two steps:

- placement ([Fet95]) and
- routing ([Wan95])

These two phases are decoupled, because during the placement the circuit is deformed step by step. This implies a manipulation of the geometrical data which would be an expensive operation, if it is performed on the final layout data. Instead the placement phase is a calculation phase, after which the positions of all objects in the circuit are known, such that the final layout can be generated by processing of these informations.

Not alike other design system for VLSI, CADIC 's layout system works hierarchically. The layout design in CADIC is also based on its DAG data structure. On the data structure, where the bicategorial expression for each treenode has been generated (c.f. figure 8.1), the system will now process each syntax tree in a bottom–up manner. The elements at the leaves of a syntax tree are basic components at the corresponding hierarchy level. The inner nodes of each tree which contain one of the operations $\ominus$ or $\oplus$ imply a local construction step for the two subtrees of this node. Mainly this construction step consists of a river routing algorithm which has to connect the borders of the two objects. The resulting wiring can be optimized according to the following aspects:

- minimization of the channel height
- minimization of the channel width
- minimization of the area for the new object which is created by the two subobjects and the channel between them

These optimization goals are respected during the placement phase, the result of which is a *placed syntax tree*. In this placed syntax tree all channel heights and sizes for the subcircuits are calculated.

Beside the representation as syntax trees, the layout design system in CADIC uses a data structure for the geometrical layout. This data structure which is called *BTG–Net* (basic topographical net), is created for each treenode in the hierarchy (c.f. figure 9.1). Again the hierarchical structure of the circuit is used, i.e. for each treenode there exists only one BTG–Net. For each instance of this treenode the corresponding BTG–Net is inserted.

9.2 The Layout for the Fulladder

In order to create the layout for a circuit select the entry `Place&Route` from the submenu `-Synthesis-` in the main menu. After you press the push button `Place&Route`, a new submenu named `Layout` is displayed in the menuline. You are now in the mode of function selection.

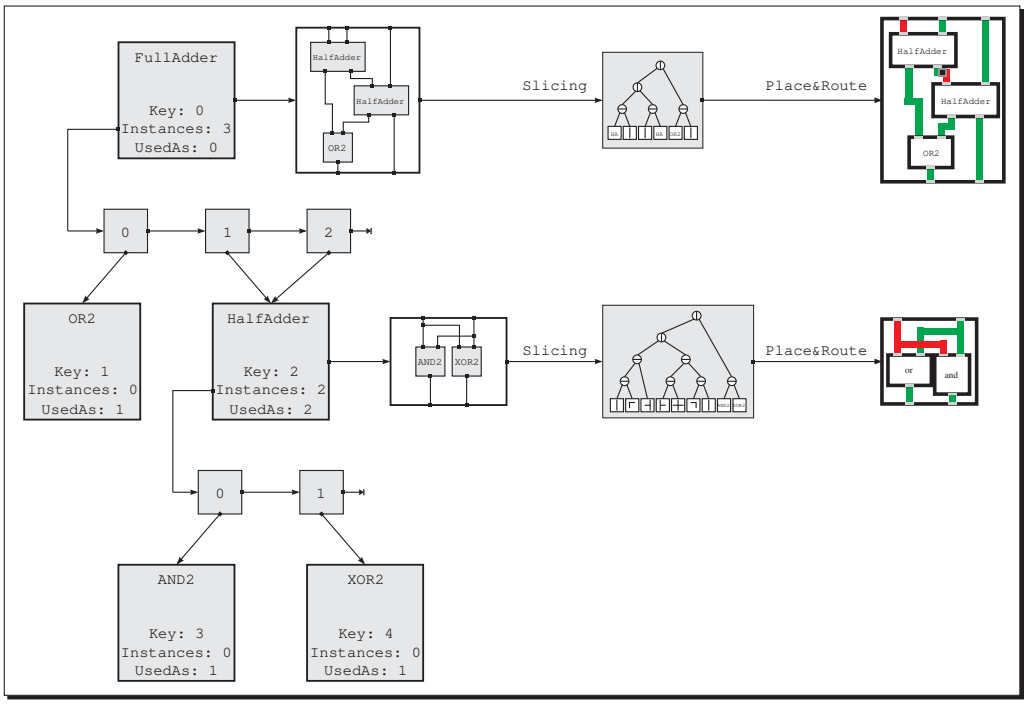
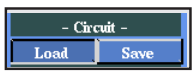


Figure 9.1: Generation of BTG–Nets for each treenode in a hierarchical description

Load Circuit



Before you can create the layout of a circuit you have to load it first. But you can use the DAG structure if you already loaded the circuit within another tool of the system, as for example the layer assignment tool. Especially you must not load it newly, if you want to use the results of a previous tool. In our case we will load again the hierarchy for `FullAdder` and create the layout without the results of the other tools. Select the push button `Load` from submenu `-Circuit-` and choose the entry `FullAdder` in the list window.

Hierarchical Placement

The function `hierarchic` from the submenu `-Placement-` will create a hierarchical geometrical layout. After you have loaded the circuit `FullAdder`, press the push button `hierarchic` within the submenu `-Placement-`. After

that the layout procedure starts up and you can read the following informations in the message window:

```
EMBED < FullAdder >
Slicing:
Generate Slicing for HalfAdder
Generate Slicing for FullAdder
Place the Slicing-Tree for FullAdder
(FullAdder(HalfAdder))
Layout-Generation:
Generate the Layout for HalfAdder
Generate the Layout for FullAdder
Data Space for BTG of FullAdder:  7940 Bytes
Data Space for BTG of HalfAdder:  5120 Bytes
Data Space for BTG of AND2:  1276 Bytes
Data Space for BTG of XOR2:  1276 Bytes
Data Space for BTG of OR2:  1276 Bytes
Total Data Space for BTG: 16888 Bytes
+----------------------------AREA-PROTOCOL----------------------------+
Lay 0:  1605.36 um^2 Lay 1:  0.00 um^2 Lay 2:  0.00 um^2 PLay:  0.00 um^2
+---------------------------------------------------------------------+
Total used area:  33605.36 um^2 Circuit Area:  58864.83 um^2 TA/CA:57.09%
+---------------------------------------------------------------------+
```

These messages illustrate the working of the place&route algorithm. First the slicing of the Cgraphs is generated in the same way as we have explained it in chapter 8. Then the slicing information is used for the placement phase, indicated by the message `Place the Slicing-Tree for FullAdder`. The placement is done in a bottom–up manner. This is indicated by the following line, where the expression `(FullAdder` indicates that the placement for `FullAdder` is started. But before it can be finished the placement for `HalfAdder` has be generated. An open bracket "`(`" indicates a step down in the hierarchy and the corresponding closing bracket "`)`" represents the backward direction. In our example the placement for `FullAdder` can be

finished after that for `HalfAdder` is finished.

After the placement is done, the system can generate the final layout of the circuit. Again this is done in a bottom–up manner. First the layout for `HalfAdder` is created, then that for `FullAdder`.

Finally the system displays information about the space used for the total layout of the circuit. Here you get the sizes for all treenodes in the hierarchy, i.e. also the sizes of the basic cells are shown, for which the layout is given within the corresponding basic cell library.

The tabular with the title `AREA-PROTOCOL` shows the size information of the circuit. In detail you can see, how much area is used for the wiring in the different layers. In our example, where we did not call the layer assignment and power supply tool, all wires are layed out in the default layer 0, such that you can read the information `Layer 0: 1605.36 um^2`. This indicates that all wires in the circuit use an area of $1605.36\mu m^2$. In the second line of the tabular the system displays the total used area of the layout, i.e. the area for the wires plus the area for the basic cells. In our case, this is $33605.36\mu m^2$. This value is put in relation to the enclosing rectangle of the circuit, such that we have a coverage of `57.09%`.

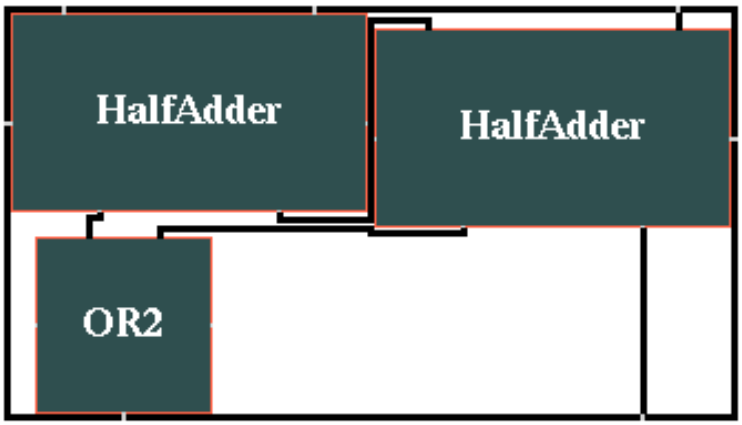


Figure 9.2: Hierarchical layout for `FullAdder`

Figure 9.2 shows the hierarchical layout for the `FullAdder`, i.e. you see the highest hierarchy level with the macro cells for the two `HalfAdder`.

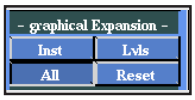
The layout differs from the schematic input of the circuit, because the used area is smaller, if the instances for the `HalfAdder` are placed beside each other. But note that the topology of the layout is the same as that for the schematic input. This means that the system does not insert new crossings of wires or totally changes the arrangement of the cells.

Graphical Expansion

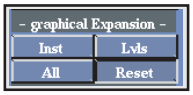
Because the display of the layout at the highest hierarchy level does not give a good impression of the structure of the circuit, you can expand the graphical representation. For the expansion of the layout the system offers you three different operations:

- expansion of a single instance
- expansion of one instance for some levels
- full expansion of the whole circuit

We will explain the first two operations later in this chapter. In the case of our small `FullAdder` example they nearly do the same as the full expansion operation.



For a complete expansion of the layout you should select the entry `All` from the submenu `-graphical Expansion-`. After the activation of this push button, the layout in the workarea is drawn again, where the two `HalfAdder` instances are replaced by their contents as you can see it in figure 9.3.

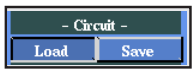


With the help of the entry `Reset` from the submenu `-graphical Expansion-` you can restore the display of the highest hierarchy level. After this first impression of the layout subsystem in CADIC we will turn to a larger example in order to explain the remaining menu functions.

Figure 9.3: Complete expansion of the layout for `FullAdder`

9.3 Layout for the 16 Bit Conditional Sum Adder

Load Circuit



Before you can create the layout of a circuit you have to load it first. But you can use the DAG structure if you already loaded the circuit within another tool of the system, as for example the layer assignment tool. Especially you must not load it newly, if you want to use the results of a previous tool. In order to load the 16 bit conditional sum adder you should select the entry `CSA[n]` from the schematic list window and type in `16` at the prompt window for the parameter values.

Hierarchical Placement

The function `hierarchic` from the submenu `-Placement-` will create a hierarchical geometrical layout. After you have created the DAG structure for `CSA[16]`, press the button `hierarchic` within the submenu `-Placement-`. After that the layout procedure starts up and you can read the following informations in the message window:

```
EMBED < CSA[16] >
Slicing:
Generate Slicing for CSA[1]
Generate Slicing for SEL[1]
```

```
Generate Slicing for SEL[2]
...
Generate Slicing for CSA[16]
Place the Slicing-Tree for CSA[16]
(CSA[16](CSA[8](CSA[4](CSA[2](CSA[1])(SEL[2](SEL[1])))(SEL[3]))
(SEL[5]))(SEL[9](SEL[4])))
Layout-Generation:
Generate the Layout for CSA[1]
Generate the Layout for SEL[1]
Generate the Layout for SEL[2]
...
Generate the Layout for CSA[16]
Data Space for BTG of CSA[16]:  66084 Bytes
Data Space for BTG of CSA[8]:  36336 Bytes
...
Total Data Space for BTG: 291192 Bytes
+----------------------------AREA-PROTOCOL----------------------------+
Lay 0:  0.00um^2 Lay 1:  112918.28um^2 Lay 2:  47372.92um^2 PLay:  0um^2
+---------------------------------------------------------------------+
Total area:  1425056.83um^2 Circuit Area:  5395939.12um^2 TA/CA:26.4%
+---------------------------------------------------------------------+
```

During layout design, many informations about the design process are displayed in the message window which include the information of slicing cells, placing the slicing-tree, layout generations for each subcircuits, and data space sizes of BTG. After the layout is successfully completed, the system will display an AREA-PROTOCOL table in the message window which gives the information about design areas. In the table, the areas for `Layer 0, Layer 1, Layer 2` and `Power-Layer` represent the areas of wires, and `Total used area` is the sum of areas of all cells and areas of all wires. `Circuit area` indicates the area used by the circuit after placement which is the enclosing rectangle region. The term `TA/CA` gives the rate of the total used area and the circuit area.

In our example we have called the layer assignment tool before the generation of the layout. Therefore in the table, the area of `Layer 0` is $0.00\mu m^2$. `Layer 1, Layer 2` are $112918.28\mu m^2$ and $47372.92\mu m^2$, repectively. The `Power-Layer` is not used. If you do not call the layer assignment tool, all the wires are embedded in the default layer 0.

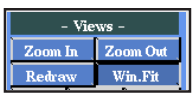
The total used area is $1425056.83\mu m^2$, and the circuit area is $5395939.12\mu m^2$. The rate between the total used area and the circuit area `TA/CA` is `26.4%`.

The complete area protocol is written in the directory defined by the environment variable `PROTDIR`. The name of the area protocol is the same as the name of the circuit and has a suffix `.areaprot`. In order to write the area protocol file, you should first set a value for the environment variable `PROTDIR`. A suggested value is `../Data/Prot`. If you do not set up the environment variable `PROTDIR`, the message window will display:

```
Environment Variable PROTDIR undefined
```

Now the hierarchical placement result of `CSA[16]` is shown in the work area. Only a part of it is visible, because the layout is too large to fit in the window in normal display mode.

Change the Display Mode



In order to see the whole layout of `CSA[16]` you can scale down the display. With the help of the entry `Win.Fit.` from the submenu `-Views-` the layout is scaled in both dimensions such that it totally fits in the workarea. The width and height are both scaled by the maximum factor.

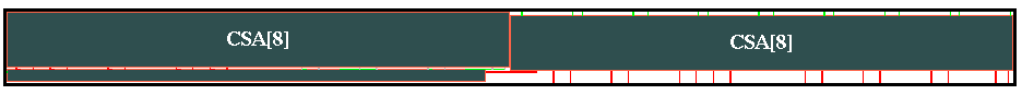
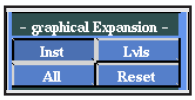


Figure 9.4: Fit the display of the layout `CSA[16]` in the workarea

9.3.1 Layout Expansion

After activating the window fit operation the layout of `CSA[16]` is displayed as it can be seen in figure 9.4. Now we will successively expand the graphical representation of this layout.

Expansion of a Single Instance



With the help of the entry `Inst` from the submenu `-graphical Expansion-` you can resolve the hierarchy level of one single instance. After you have activated this menu point you are in the instance selection mode. If you move the pointer into the workarea, the nearest instance in the layout will be highlighted, i.e. its border changes from red to cyan. Now press the left mouse button and the instance will be replaced by its contained subcircuits.

In figure 9.5 we have selected the right instance `CSA[8]` and substituted by the following hierarchy level. This operation illustrates the recursive description of the conditional sum adder, because `CSA[8]` is built by the same components as `CSA[16]`, but with smaller parameter values and therefore smaller layout areas.

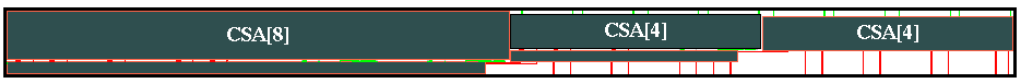
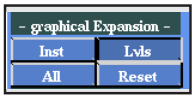


Figure 9.5: Expanding the right instance `CSA[8]` one single step

After the expansion you remain in the instance selection mode in order to expand more instances. Note that now the subcircuits from the just expanded instances are also selectable, because they are at the same level now. You can abort the selection mode by pressing the right mouse button within the workarea.

Expansion over more Levels



While you could perform a detailed expansion of the layout with the help of the previously explained function `Inst`, this operation is tiresome, if you want to expand the layout down to the lower levels. For this purpose CADIC

offers you the entry `Lvls` from the submenu `-graphical Expansion-`.

After the activation of this menu item you are in instance selection mode again in order to choose the candidate for the expansion operation. Pick the highlighted instance by pressing the left mouse button. Now an input window will popup on the workarea asking you the number of levels, for which the instance should be expanded:

```
Please enter the Depth of Expansion:  <3    >
```

In figure 9.6 we have expanded the left instance `CSA[8]` for three hierarchy levels, i.e. the instances you see at the top of the layout are 1 bit adders `CSA[1]`.

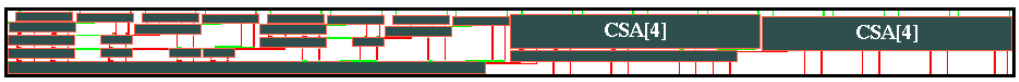


Figure 9.6: Expanding the left instance `CSA[8]` for three hierarchy levels

After the instance has been expanded you are still in instance selection mode which can be aborted by pressing the right mouse button in the workarea. You can use the other expansion operation on an already expanded layout, i.e. you can first expand one instance for some levels, then pick single instances and expand them step by step, etc.

With the help of the function `Lvls` you can expand one single instance down to the basic cell level. This is done, if you enter a depth for the expansion which is larger than the lowest level number. An easier way is to enter `-1`, especially if you do not know, how much hierarchy levels are contained in the corresponding instance.

Full Expansion of the Layout

As shown for the example of the `FullAdder` we can completely expand the layout with the help of the entry `All` from the submenu `-graphical Expansion-`. If you choose this menu item, the layout is fully expanded down to the level of basic cells as you can see it in figure 9.7.

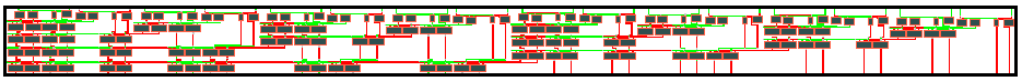
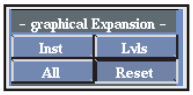


Figure 9.7: Complete expansion of the layout for `CSA[16]`



With the help of the entry `Reset` from the submenu `-graphical Expansion-` you can restore the display of the highest hierarchy level, i.e. the system will display the layout in the form of figure ??, because you previously activated the window fit display mode.

9.3.2 Layout Tracing

You can hierarchically trace your layout design by using the function `Tracing`. As in the case of the Cgraph structure there are two directions to trace the layout: `Down` and `Up`. We still take the circuit `CSA[16]` as an example to explain this function. For these tracing functions yon can refer to the Section 5.6 for references.

Tracing Down

Because you are now in the highest level of the layout, you can use the tracing down function only. If you select the entry `Down` in the submenu `-Tracing-`, you are in instance selection mode. Move the cursor further to the subcircuit that you want to trace into, as for example the right instance `CSA[8]`, and make its border highlight in blue. Now press the left mouse button and the hierarchical layout of `CSA[8]` appears in the work area replacing that of `CSA[16]` as shown in figure 9.8.

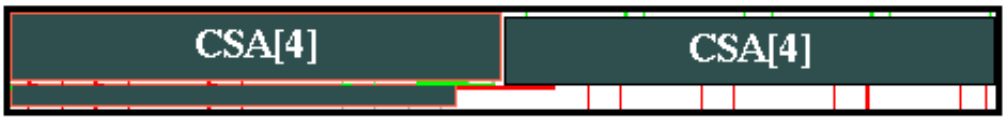


Figure 9.8: Tracing down to the layout of `CSA[8]`

Note: You can not use the expansion functions from the submenu `-graphical Expansion-` during the tracing of the layout. For example,

if you use the function `Inst` within the above `CSA[8]`, you will see the following warning the message window:

```
WARNING: no expansion while tracing!
```

If you want to exit the tracing down function, move the pointer to the work area and press the left mouse button. Then, you return to the menu selection mode again. If you are in the lowest level of the hierarchical layout, you can also press the left mouse button to quit the function of tracing down and return to the menu selection mode again.

At each level of the hierarchy you can enlarge the display of the layout by selecting the entry `Win.Fit` from the submenu `-Views-`. Then the current hierarchy level is scaled, such that it fits in the workarea. This is very helpful, if you trace down to the lower hierarchy levels, where the areas of the components are very small.

Tracing Up

If you are not at the highest level of the layout hierarchy, you can use the tracing up function to go back to higher levels. Each time you select the entry `Up` from the submenu `-Tracing-` you return to the next higher level of your tracing path. Thus, you can repeatedly press the push button `Up` until you are at the desired level.

9.3.3 Layout Views

The view functions consist of the zoom functions, the redraw function and the window fit function which we have already mentioned above. You can use the functions `Zoom In` and `Zoom Out` to enlarge or shrink the current display of the layout. The functions `Redraw` and `Win.Fit` can be used to adjust the display size to fit in the work area.

Zoom In

In order to enlarge the current display in the workarea, you can select the entry `Zoom In` in the submenu `-Views-`. Move the pointer into the workarea

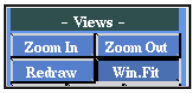
and you can locate the start position of the zoom window. After you have fixed the start point by pressing the left mouse button you can drag a rectangle frame in order to setup the area of the layout which will be scaled to fill the whole workarea. Locate the opposite corner of the rectangle by pressing the left mouse button again. Then, the chosen part is enlarged and appears in the work area to replace the former display. After that you automatically return to the menu selection mode again.

Zoom Out

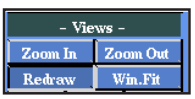
By using this function you can shrink the current display that has been enlarged. Press the push button `Zoom Out` in the submenu `-Views-`, then you return to the previous display mode, i.e. all selected areas during the zooming operations are pushed onto a stack. Each zoom out operation pops the upmost frame from the stack and restores the corresponding display of the layout.

Note: Only after you have enlarged a display, you can use the function `Zoom Out` to shrink it again. If there are no frames on the stack, the function `Zoom Out` will have no effects.

Redraw and Window Fit

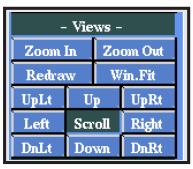


In general the display of a layout has not always a just suitable size for the workarea. For example, the initial display of the layout for `CSA[16]` shows only a part of the whole layout. In order to display the layout fully within the workarea you can select the entry `Win.Fit` in the submenu `-Views-` as we have already shown it in the previous sections. Figure ?? shows the result after executing the function `Win.Fit` for `CSA[16]`.



If you want to return to the initial display, you can use this function. By pressing the push button `Redraw` in the submenu `-Views-`, the initial display will appear in the workarea to replace the current display.

9.3.4 Layout Scrolling



You can scroll the display of the layout result in the work area by using the scroll functions. The scroll functions consist of eight directions: use the function `UpLt` to scroll directly to the upper left corner, `Up` to scroll up one screen, `UpRt` to the upper right corner, `Left` to the left, `Right` to the right, `DnLt` to the lower left corner, `Down` to scroll down one screen and `DnRt` to the lower right corner. If you want to use one of these functions, press the corresponding push button in the submenu `-View-`. You can repeatedly use the same function by pressing the corresponding push button many times until you obtain the desired position.

9.3.5 Output of the Layout Result

By using the output function you can obtain a postscript file of your layout design. Suppose that you want to output the layout result of `CSA[16]`. Move the cursor to the push button `PostScript` in the submenu `-Output-`, and press the left mouse button. Now, a input window pops up onto the work area with the following prompt:

```
Please enter Scaling Factor:  <1    >
```

The scaling factor must be a positive number. If you do not input any number, but direct press the `Return` key, the input window will disappear from the work area and abort the output function. Here we choose the scaling factor `1` first and press the `Return` key. Then, a prompt appears in the input window as follows:

```
Circuit needs 02×10 pages.  Accept (y/n)?:  <n>
```

It means that the circuit will be output in the size of twenty A4 pages. If you accept this size, type `y` and press the `Return` key. If you do not accept this size, type `n` and press the `Return` key. Now, the input window shows the prompt again:

```
Please enter Scaling Factor:  <0.25   >
```

Now we select the scaling factor `0.25` and press the `Return` key. After that, the following message appears in the input window:

`Circuit needs 01×03 pages. Accept (y/n)?: <y>`

It means that the circuit will be output in the size of three A4 pages. If you want to accept this size, you can type `y`. Otherwise, type `n` in order to select another output size.

After you type `y` and press the `Return` key, the layout that is currently displayed in the workarea will be output and stored in a file with the suffix `ps`, such as `CSA[16].ps` within the current directory.