7

Layer Assignment

7.1 Introduction

During the specification of a circuit we have neglected the layers, in which the wires are embedded, in order to support a comfortable description method. It is the task of a design tool to assign the signal nets to the given wiring layers (*layer assignment problem*). Wiring segments which are intersecting and which do not belong to the same signal net, must be embedded in two different layers. Parts of a signal net which are on different layers, must be connected by vias. Figure 7.1 shows a subcircuit which is embedded in two layers. In this case we need two vias to get a legal layer assignment.

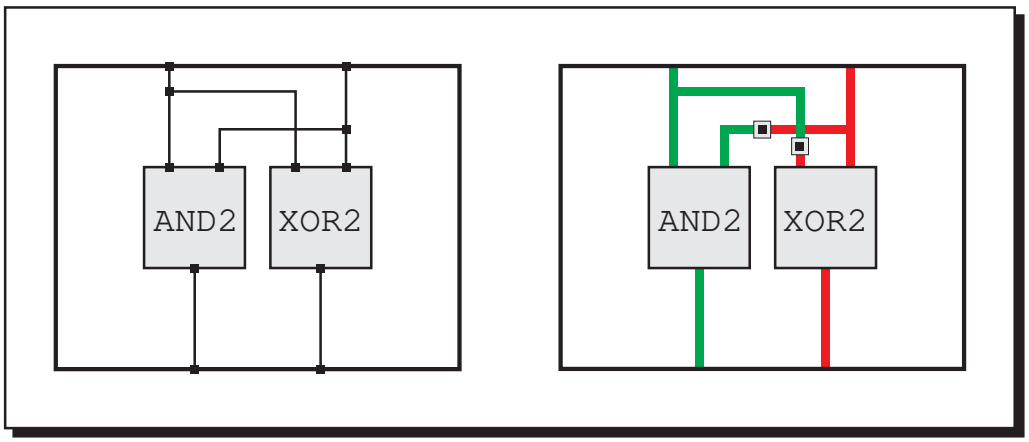


Figure 7.1: Circuit with layer assignment in two layers

An important aspect of algorithms for automatic generation of a layer assignment lies in the minimization of the number of vias. The reason for this is that vias delay the signals and they cause errors during the production process of the chip.

This optimization problem is called *constrained via minimization problem* (CVM), where we use n different layers in the general case (CVM_n). But industrial chip fabrication uses only two layers for the logical wiring of the chip and one additional metal layer for the supply nets. During the assignment of the signal nets to the layers, there may be some constraints which have to be taken into account. For example if you have layers with different conductivity, you should use the layer with higher conductivity for most parts of the signal nets (*layer assignment problem with layer preference*). Another constraint can be given by the basic cells, if you may only connect a pin in a certain layer (*layer assignment problem with pin preassignment*). A summary about the theory of layer assignment problems which in general are very difficult to solve, can be found in [Mol87],[CNR87].

7.2 The Algorithm integrated in CADIC

In the following we restrict to an algorithm which solves the problem CVM_2, where vias may only be set on wiring segments and not on branches or knees. In [KCS88] it is shown that there exists a polynomial algorithm for this restricted problem.

The computation of the layer assignment is done by a hierarchically working algorithm on the DAG structure of the circuit, i.e. each treenode in the hierarchy is treated only once. The algorithm works bottom up, i.e. the layers for a treenode are calculated, if they exist for all its instances. For the basic cells, the layers, in which the pins may be connected, are stored in the basic library. For each hierarchy level we have to solve the layer assignment problem with pin preassignment, because from the lower level the layers at the borders of the instances are implied. An optimal solution for this problem in polynomial time is unknown until now ([KMO89]).

In a simple approach the algorithm could be run without regarding the given preassignment of the pins. Afterwards we could adjust the layers at the instance borders by creating appropriate vias. It is obvious that this method would lead to too many vias. Therefore we hold for each Cgraph two different layer assignments which are "dual" to each other. The second version of the Cgraph is given by exchanging the layer for all signal nets, i.e. a net in the red layer in the normal version is in the green layer in the dual version and vice versa. From these two possibilities we choose this one, where fewest vias are needed to adjust the instance border to the surrounding Cgraph.

The integrated tool in CADIC ([Gra92]) is based on the algorithm described in [Mol93]. This algorithm works on a Cgraph as input, for which it calculates the dual graph first. The faces of the Cgraph which are represented by the nodes of the dual graph, are marked as "odd" or "even". This classification follows directly from a classification for the nodes at the border of each face. The edges of the dual graph represent the neighbourhood of two faces in the Cgraph, i.e. these faces have a common border. In [Mol93] it is shown that there is a legal layer assignment for the Cgraph, if for each pair of odd faces there exists a path within the dual graph. The edges on this path directly imply the locations of the vias. It is also shown in [Mol93] that such a "marriage" of the odd faces is always possible, because there is always an even number of odd faces in a given Cgraph. The problem of minimizing the number of vias for a legal layer assignment is now equivalent to finding shortest paths for the connection of two odd faces.

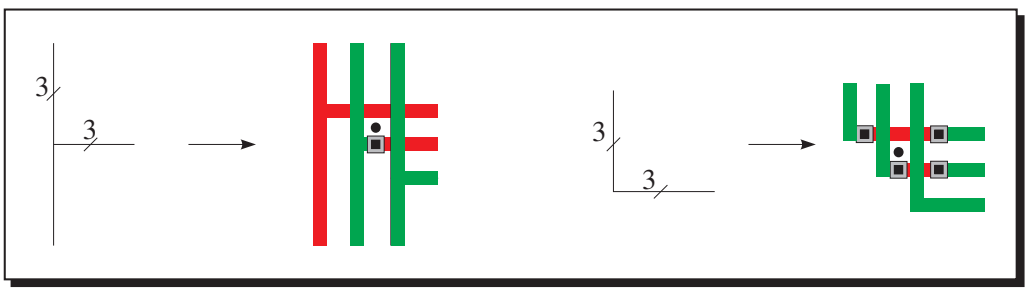


Figure 7.2: Layer assignment for multi wire nodes

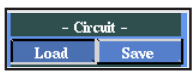
For a legal layer assignment we still have to consider the width of wires, because the branching or crossing of wires with a width larger than one implies more vias as figure 7.2 illustrates. The vias are needed, because the refinement of the multi wire node contains odd faces (marked by •). In order to reduce the number of vias we allow the use of *multi layer busses*, i.e. not all wires have to be in the same layer. In the same way a *bus via* represents a set of vias on the single wires of the bus.

7.3 Layer Assignment for the Halfadder

In order to invoke the layer assignment tool you first should select the entry `Layer Assignment` from the submenu `-Synthesis-` within the main menu.

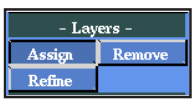
To show you how the system will display the information from the layer assignment algorithm we will first apply it to the simple design of the halfadder.

Load Circuit Hierarchy



First you have to load the DAG structure of the circuit `HalfAdder`. This can be done by selecting the entry `Load` from the submenu `-Circuit-` which calls the same function as the load button from the hierarchy menu (c.f. chapter 5). If you still have loaded the circuit `HalfAdder` you need not select this menu point.

Assign the Layers



To compute the layer assignment for `HalfAdder` you should activate the entry `Assign` from the submenu `-Layers-`. In the message window of CADIC you can read the following messages:

```
building_dualgraph HalfAdder
number of odd_faces 2
odd_face_marriage in dualgraph HalfAdder
inserting_layer_information HalfAdder
number of created via's:  2
```

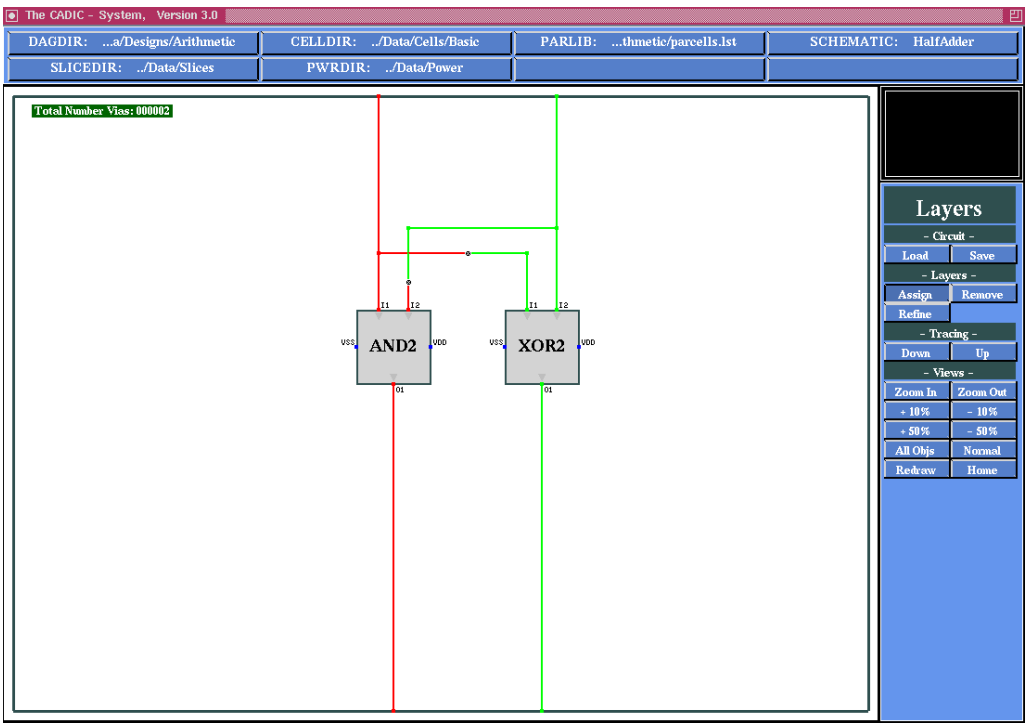


Figure 7.3: Layer assignment for `HalfAdder`

The first message shows that the system builds the dual graph from the Cgraph of `HalfAdder` and that it contains 2 odd faces. In the next step these faces are "married", i.e. the shortest path from one face to the other is calculated in the dual graph. This path implies the creation of two vias, i.e. the path consists of two edges in the dual graph.

After the last message is displayed the graphical representation of `HalfAdder` is changed in the workarea (c.f. figure 7.3). The wires which have been coloured yellow, are now drawn in red and green. These two colours represent the two different wiring layers, whereas yellow denotes the unspecified layer.

In the upper left corner of the workarea you can read a label `Total Number Vias: 000002` which indicates that for the whole circuit there have been created two vias. In this example of one single hierarchy level this number is equivalent to the vias on this level.

The vias are denoted by small squares on the wires in the Cgraph. At

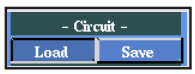
these positions the layer for the signal net changes from red to green. The algorithm puts the vias always in the middle of the corresponding Cgraph edge.

If you look at figure 7.3 you might ask yourself, why are the vias needed. We could layout the wire at the second input `I2` of the `AND2` gate in the green layer and the wire connected to the left input `I1` of the `XOR2` gate in the red layer. The reason for the vias is that the algorithm solves the layer assignment problem with pin preassignment. In our case the pins of the basic cells are preassigned within the basic cell library, such that all pins are either in the red layer or in the green layer.

7.4 Colouring the Fulladder

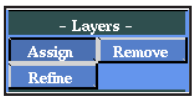
Now we apply the layer assignment algorithm to a small hierarchical circuit description.

Load Circuit Hierarchy



In the same way as above you should load the DAG structure for `FullAdder` by selecting the appropriate entry from the list window.

Assign the Layers



After the DAG structure has been built we can now compute the layer assignment for `FullAdder`. As shown above you can do this by activating the entry `Assign` from the submenu `-Layers-`. In the message window the system will now display the following informations:

```
building_dualgraph HalfAdder
number of odd_faces 2
odd_face_marriage in dualgraph HalfAdder
inserting_layer_information HalfAdder
building_dualgraph FullAdder
number of odd_faces 2
odd_face_marriage in dualgraph FullAdder
```

```
inserting_layer_information FullAdder
number of created via's:  5
```

In this case of a hierarchical circuit the system has to build the dual graphs for all treenodes, i.e. for `HalfAdder` and `FullAdder`. As mentioned above the algorithm works bottom–up, such that `HalfAdder` is treated first and the layer assignment for it is used to compute the layer assignment for the Cgraph of `FullAdder`. Both Cgraphs have two odd faces, which have to be "married". As we know from the previous section the algorithm needs two vias for a legal layer assignment of `HalfAdder`. One additional via is needed within the Cgraph of `FullAdder`, such that we have a total number of 5 vias within the whole hierarchy of the circuit.

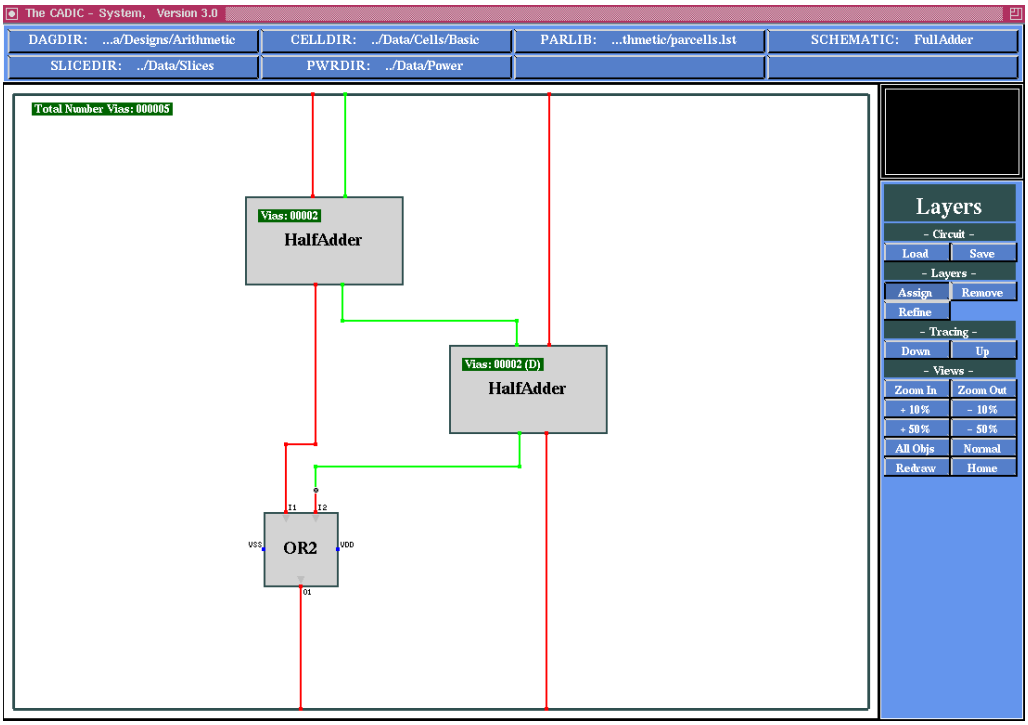


Figure 7.4: Layer assignment for `FullAdder`

In figure 7.4 the root level of the hierarchy for `FullAdder` is shown. If you look at the two instances `HalfAdder`, you can see that for the left instance the sequence of layers at the northern and southern border is (red, green). For the right `HalfAdder` the sequence is (green, red), i.e. for the second

instance we use the dual layer assignment, where the red and green layers are exchanged. This helps the algorithm to minimize the number of vias. If it had to use the same layer assignment for both instances, then it must create a via on the wire connecting the right output of the left `HalfAdder` with the first input of the right `HalfAdder`.

In each instance which represents a subcircuit, you can read the number of vias created in the corresponding DAG structure. Here both instances of `HalfAdder` contain two vias from the layer assignment of the previous section. The annotation `(D)` behind the number of vias in the label of the right `HalfAdder` indicates that for this instance the algorithm has chosen the dual layer assignment.

Tracing through the Results

We can control the layering of the `HalfAdder` with the help of the tracing functions from chapter 5. First select the entry `Down` from the submenu `-Tracing-` in order to descend into one halfadder. Move the pointer into the workarea near to the left instance and select it with the left mouse button. Now the system descends one level in the hierarchy and displays the Cgraph for the `HalfAdder`. Because the instance is in normal mode, you see the same layer assignment as we have created it in the previous section (c.f. figure 7.3).

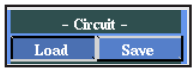
In the next step we want to look at the second `HalfAdder`. To do this, you must terminate the cell selection mode by pressing the right mouse button within the workarea. After that we trace up one level back to the `FullAdder` by selecting the entry `Up` from the submenu `-Tracing-`. Now you can step down into the second `HalfAdder` with the trace down operation. Compared to the first `HalfAdder` you will see that the layer assignment is inverted.

7.5 Multi Layer Wires: the 16 Bit Conditional Sum Adder

For the layer assignment algorithm the example of the 2^n bit comparison tree `Tree[n]` is an "uninteresting" circuit, because there are no crossings of wires in the tree structure. All wires can be layed out in the same layer, such that there are no vias needed.

We will now examine the results for the 16 bit conditional sum adder and especially show how bundles of wires may be assigned to the different layers.

Load Circuit Hierarchy

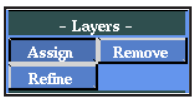


Load the 16 bit adder from our parameterized specification of the n bit conditional sum adder. As you already know from chapter 5 you have to activate the entry `Load` from the submenu `-Circuit-` and select the element `CSA[n]` from the list window. At the following prompt

```
Enter Parameter Values for "CSA[n]":<      >
```

type in `CSA[16]` or simply `16` to set the value of `n`=16. After confirming this entry with the return key the system will load the DAG structure for the 16 bit adder.

Assign the Layers



Now you can call the layer assignment algorithm for `CSA[16]` by activating the entry `Assign` from the submenu `-Layers-`. Following the hierarchical DAG structure, the algorithm displays a list of messages:

```
building_dualgraph CSA[1]
number of odd_faces 4
odd_face_marriage in dualgraph CSA[1]
inserting_layer_information CSA[1]
building_dualgraph SEL[1]
number of odd_faces 6
odd_face_marriage in dualgraph SEL[1]
```

```
inserting_layer_information SEL[1]
...
building_dualgraph CSA[16]
number of odd_faces 0
odd_face_marriage in dualgraph CSA[16]
inserting_layer_information CSA[16]
number of created via's:  402
```

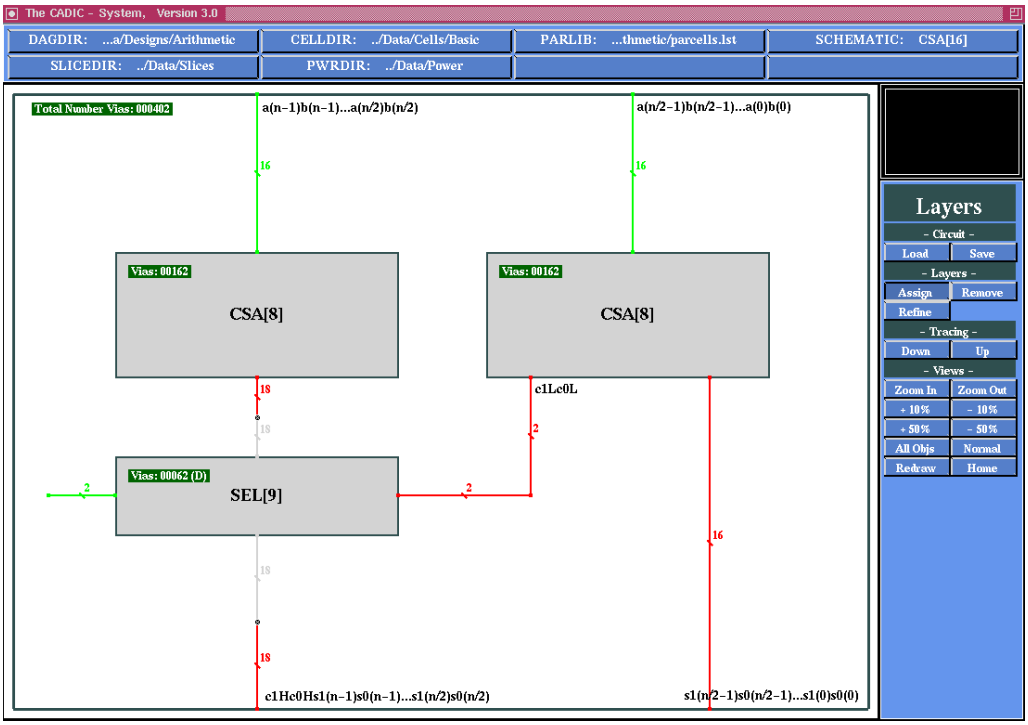


Figure 7.5: Layer assignment for the 16 bit conditional sum adder

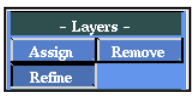
Figure 7.5 shows the result of the layer assignment algorithm for the topmost hierarchy level of `CSA[16]`. The label in the upleft corner of the workarea tells you that there have been inserted 402 via nodes in all Cgraphs of the hierarchy. The informations in the instances show, how these vias are split across the lower levels. Each `CSA[8]` contains 162 vias and the selection subcircuit `SEL[9]` needs 62 vias. If you sum up these numbers, you get $2 \times 162 + 62 = 386$. This means that the remaining 16 vias are on the topmost hierachy level. But in figure 7.5 you will only find two via nodes which are placed on the wire connecting the southern border of the left `CSA[8]` and

the northern border of `SEL[9]` and the wire between the southern border of `SEL[9]` and the southern border of the hierarchy level. These two via nodes represent not only one single via, because they are put onto a bundle of wires. As you can see in figure 7.6 each via node corresponds to 8 single vias on this bus wire.

The presence of a multi via node can be recognized by the change of the colours for a bus wire. In the example from figure 7.5 the colour of the bus between `CSA[8]` and `SEL[9]` changes from red to grey. This means that the bus is first completely layed out in the red layer and below the multi via node some single wires of the bus are in the green layer. The bus is then drawn in grey to denote that it is a multi layer wire. In the same way the bus wire at the southern border of `SEL[9]` is a multi layer wire. The remaining bus wires are all layed out in one layer, such as the two wires at the northern border of the `CSA[8]` instances are totally in the green layer.

You might wonder, why the bus via between the southern border of `SEL[9]` and the schematic border is needed. The multi layer wire could be directly connected to the border without changing the layers again. But one property of the underlying algorithm is that a bus wire must be uniformly connected to the schematic border in order to simplify the reuse of each Cgraph as an instance in another hierarchy level. If a multi layer wire has to be connected to the Cgraph border it is unified, such that the minimal number of vias is inserted. In our example the wire is totally layed out in the red layer, because the algorithm needs only 8 vias instead of 10 vias for a uniform wire in the green layer.

Refining Multi Layer Wires



If you want a detailed display of a multi layer wire node, you can select the entry `Refine` from the submenu `-Layers-`. After activating this function you are in wire selection mode. Move the pointer into the workarea near to the bus wire you want to examine. If the desired wire is highlighted (its colour changes to cyan) you can select it by pressing the left mouse button.

After that a small window is popped up upon the bus, in which you see the colours of all single wires. In figure 7.6 we have selected the multi layer wire at the southern border of the instance `SEL[9]` and refined it.

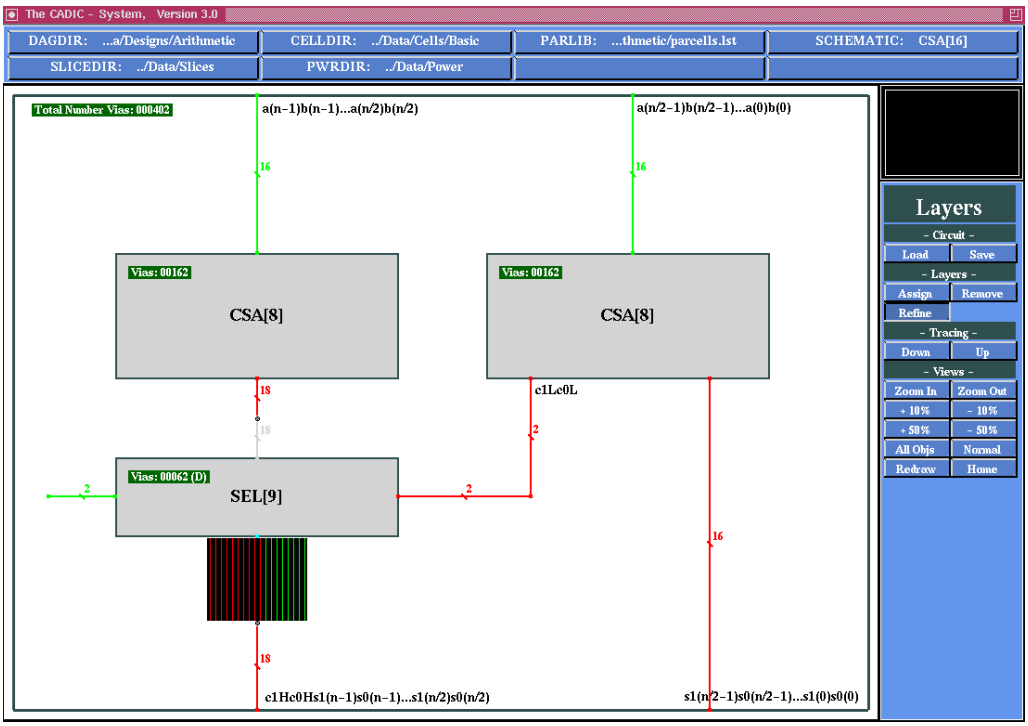


Figure 7.6: Refinement operation for multi layer wires

From the refinement you can also find out, where the vias of a bus via are placed. In figure 7.6 the vias are placed on the last 8 wires of the 18 bit wide bus, because in the bus refinement the last 8 wires are layed out in the green layer and after the bus via the whole wire is in the red layer.

After the popup window is opened in the workarea you remain in wire selection mode in order to refine another bus. For example move the pointer near to the other grey wire between `CSA[8]` and `SEL[9]`. If it is highlighted in cyan, press the left mouse button to select this one. Then the previous popup window disappears and a new window is creted at the position of the new selection. You can terminate the wire selection mode by pressing the right mouse button within the workarea. The current popup window is closed and you return to the menu selection mode.

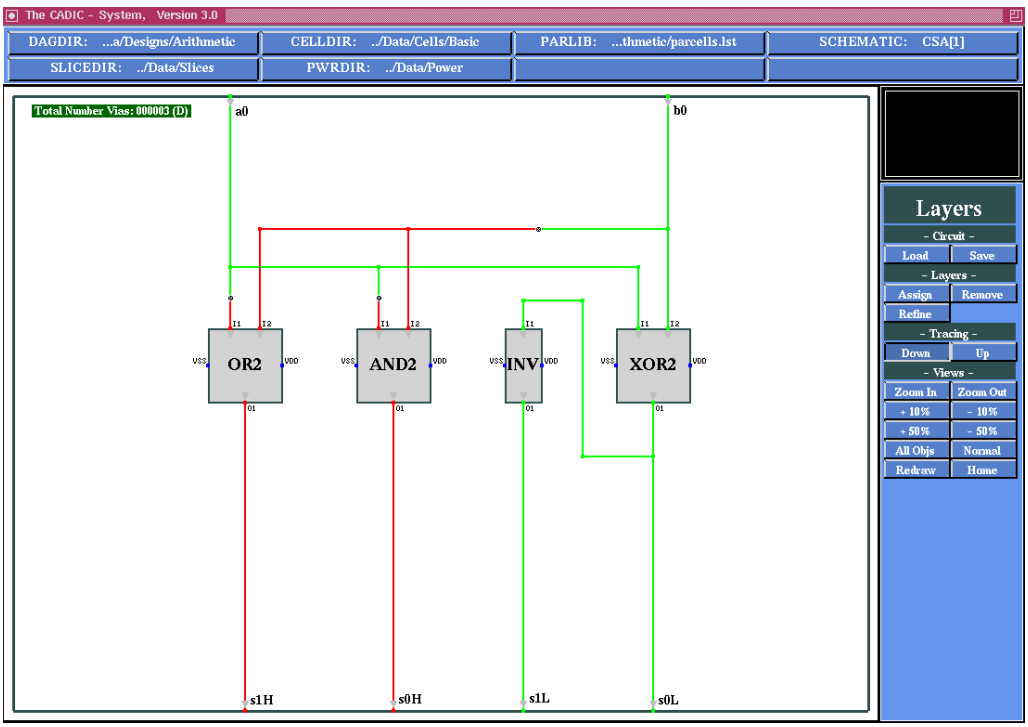


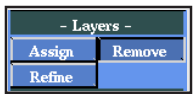
Figure 7.7: Navigation to the lowest hierarchy level `CSA[1]`

Navigation through the Results

Before we close the layer assignment session we want to take a look at the lower hierarchy levels of `CSA[16]`. Select the entry `Down` from the submenu `-Tracing-` in order to descend into the instances. Move the pointer near to the left `CSA[8]` and trace down this path until you arrive at the lowest hierarchy level `CSA[1]`. The layer assignment for the corresponding Cgraph is shown in figure 7.7.

The label in the upleft corner of `CSA[1]` tells you that there are three vias inserted and that the currently selected instance is in dual mode `(D)`. This is respected in the colouring of the wires, where the green and red layers are exchanged. You can notice this by first tracing up one level and then stepping down into the second `CSA[1]` which is in normal mode.

Removing the Layer Assignment



If you want to apply other design tools, where you do not use the layer assignment, you can remove the via nodes and the layer information from the Cgraphs. You also can load the circuit again from `DAGDIR`. To remove the layer assignment select the entry `Remove` from the submenu `-Layers-`. After activating this function the system will display the following informations in the message window:

```
2 Vias (Nodes/Edges) removed in Treenode CSA[16]
2 Vias (Nodes/Edges) removed in Treenode CSA[8]
3 Vias (Nodes/Edges) removed in Treenode CSA[4]
3 Vias (Nodes/Edges) removed in Treenode CSA[2]
3 Vias (Nodes/Edges) removed in Treenode CSA[1]
0 Vias (Nodes/Edges) removed in Treenode OR2
0 Vias (Nodes/Edges) removed in Treenode AND2
0 Vias (Nodes/Edges) removed in Treenode INV
0 Vias (Nodes/Edges) removed in Treenode XOR2
2 Vias (Nodes/Edges) removed in Treenode SEL[2]
3 Vias (Nodes/Edges) removed in Treenode SEL[1]
0 Vias (Nodes/Edges) removed in Treenode MUX
3 Vias (Nodes/Edges) removed in Treenode SEL[3]
3 Vias (Nodes/Edges) removed in Treenode SEL[5]
2 Vias (Nodes/Edges) removed in Treenode SEL[9]
4 Vias (Nodes/Edges) removed in Treenode SEL[4]
Total Number of removed Vias:  30
```

The number of removed vias concerns the via nodes, i.e. a bus via is regarded as one single node in this listing. For example at the hierarchy level `CSA[16]` there are only 2 vias removed which are exactly the two bus vias from above. Each of these two nodes represents 8 single vias as we have shown in the previous paragraph.

Note that the layer assignment is removed from the currently displayed hierarchy level down to the leaves of the DAG structure. If you want to

remove it for the whole DAG structure, then you have to trace up first, until you reach the root of the hierarchy.

Changing the View

With the help of the functions from the submenu `-Views-` you can change the graphical representation of the schematic. The functions in the menu of the layer assignment tool are identically to those in the hierarchy menu. For an explanation you can look at section 5.9.

To terminate the layer assignment menu you have to terminate any currently active 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 close the layer assignment menu by pressing the right mouse button within the menuline. Now 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.

Normally the layer assignment and the power supply tools are called before you create the final layout of the circuit. When you have created the layer assignment for a circuit you can change to the menu for generating its layout by going to the main menu of CADIC and then to the layout submenu (c.f. chapter 9).