CHAPTER 5

Physical Design: standard-cells based layout

Here we are, ready to generate the layout of our designs and to check their signal integrity criticalities. The tool we are going to use is **Encounter** by Cadence: the leader in the physical design CAD market.

Generate a new directory cap5. Copy all the files from */home/mariagrazia.graziano/ms/cap5/*
To set correctly the environment variables type:

prompt> **setedi**

5.1 Physical design of the Ripple Carry Adder

Your first exercise will be to work on a simple SUM structure organized as in figure 5.1 based on a registered 128 bit RCA (the same you used in a previous lab). You have the vhdl files, and, as Encounter starts from a verilog netlist, you have the verilog file as well **sum.v**.
You also have in this directory the synthesis constraint script file (sum-t.scr, read it and notice the command used and the period) and the resulting timing and power performance obtained AFTER the synthesis: read them and notice the requested period and frequency (sum_timeopt_t.rpt and sum_timeopt_pw.rpt). We will compare these values to those obtained after the layout phase.

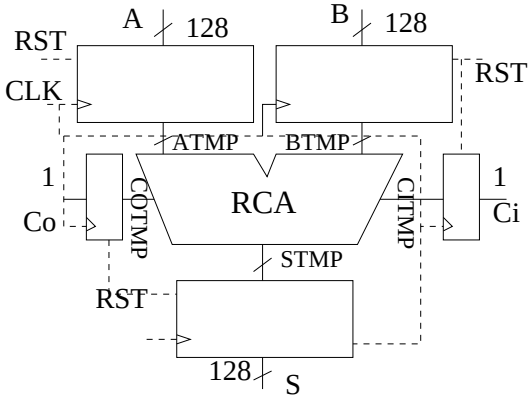

Figure 5.1:

We are now ready to start. In the shell in which you set the environment variables type:

prompt> **encounter -simplefont**

Pay attention: this shell is now a dialog shell in which it is better if you do not write anything: just read it the results each time you give a command in the gui.

You will be helped step by step in the following. Anyway, you can also read Encounter manual

by typing: **helpenc** in the command line. The online help (top right box) is also a useful in case you want to know details on commands.

We are going to process the architecture physical design step by step:

Configuring→Floorplanning→Power planning and routing →Cell placing → Signal routing

Configuring The first step basically helps encounter finding the path to the libraries used for the desing and reading the synthesized version of the circuit.
From the top encounter menu import the configuration file which sets the correct references to the libraries: **File→ InportDesign**; a new window opens: select in the menu below **Load** and select from the browse menu in the new windos the file **SUM.globals**. Notice the definition of the verilog file (.v) containing the post synthesis cell view, the reference to the layout view of the library of cells (file .lef) and the definition of vdd and gnd power supply names. Notice also the Default.view file included. This stores information on the MMMC (MultiMode MultiCorner) analysis, i.e. references to files and data definitions organized and combined in order to consider not only default conditions but best case or worst case conditions in terms of temperature and of process variations. You can click on the Create Analysis Configuration button to inspect the conditions defined and read the comments in the wizard window (do not change the configurations). When done and when you have closed this wizard, click **Open** OK in the Design Inport window bottom menu: the design is imported with the correct cell reference and with the RTL verilog description.
If you press "f" a square with several lines will appear. This image represents a rougth structure where several cells will be placed in this sequence of lines.

Structuring the Floorplan At this step Encounter sets the area to be assigned to the cell ensemble and the area where the power supply ring will be routed as shown in figure 5.1.

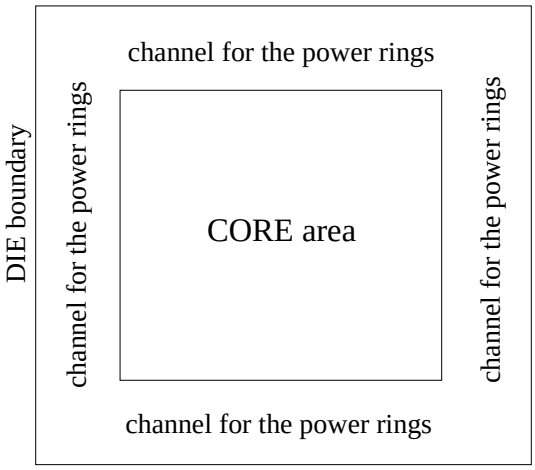

Figure 5.2: Area of the die dedicated to the core (the place for the cells) and to the power supply rings all around the core

From the Main window top menu select: **Floorplan → Specify Floorplan**. An option window opens, allowing you to define the Core-aspect-ratio (default), the channel to be used for ring routing: select Core-to-die-boundary, and force 4 (m) from the four sides of the core (unity is m here). From the top menu select now **Advanced** Note the standard cell row style: what does it allow to do (ask the teacher)? Click OK and see what happens. The cells height is already known at this point, as you can note by the gray horizontal lines. All the standards cells have the same height by construction. The width changes coherently with the transistors and interconnections area. At this point encounter knows how many rows will be needed for the design.

Inserting power Rings The channel defined before will be filled by two metal rings for power and ground respectively. These metal stripes will be connected with the VDD and GND pads (if this is the final chip a wirebonding package is supposed here, if not, more probable, these rings will be connected to other rings of other blocks) and will distribute hierarchically the power and ground signal to the whole chip. For this reason we will use for the rings an high metal layer. For avoiding congestion we will choose a different layer for the horizontal lines (M9) and for the vertical lines (M10).

From the Main window top menu select: **Power → Power Planning → Add Rings**. An option window opens, allowing you to define:

- the nets to be associated to power and ground: for this click on the browse buttom on the right end of the Net box, select and add both *vdd* and *gnd*
- the type of ring: in this case leave the default configuration as "Around core boundaries"
- the metal associated to the rings: use M9 for Top and Bottom lines, and M10 for left and right lines; leav default widths and spacing

Click OK. Now two rings have been designed: one for VDD and one for GND (the connection to the electrical generator will be done later). Note that in the corners there are vias for the connection between M9 and M10. If you click on one of the stripes you will receive information in the botton window on the metal layer, on the stripe geometry and on its cohordinates. What does this rectangle rapresent for the foundry that is going to produce your chip?

Inserting stripes. In this step vertical metal wires will be added connecting the ring from top to bottom, and will be still VDD and GND lines. This step could be avoided, but is helps in correctly distributing the power and ground signals to the chip center, so it is generally performed. The higher is the vertical stripes number, the better will be the distribution, but the more problematic will be congestion when the other signals will be routed.

From the Main window top menu select: **Power→Power Planning→ Add Stripes**. Select in the new window: for the Nets, click on the brows button and select adding vdd and gnd; M10 for routing with the default width value for vertical stripes, select **Set-to-set-distances** and choose a value of 20, in the **Relative-from-core** choose 15 for the left distance; leave the right distance unconstrained.
As you can see three sets of stripes have been routed. Each set is composed of one stripe connected to GND (click on it to see the properties) and one stripe used as a VDD line. Use the ruler (left menu) to check distances.

Standard Cell Power routing. This operation allows to place horizontal wires preparing the VDD and GND wires for the standard cells. Such wires will be connected to the ring and to the vertical stripes as well.

From the Main window top menu select: **Route→Special Route**, select *vdd* and *gnd* as nets and click OK with the default values.
Once the operation is concluded you can reckon the metal layer and connectivity by clicking the arrow in the left menu, sweeping on the metal stripes and reading in the bottom left window the metal lines attributes.

Placement. Now the cells will be placed. Up to this point the only thing known was the total area of the circuit and the number of rows to be used. After this point the layout of each cell has a unique position in one of the predefined rows.
Before starting the placement phase select **Place→Specify→Placement Blockage** and select layers from M1 to M8, such that the cells are placed out from the space occupied by the power and ground stripes, just to avoid congestion problems. Now click on **Place→Place Standard Cell**, accept the medium effort window and select to save the placement view. Click OK. What happens?
You can select the cells to read (bottom left window) the name, and zoom in the view for

recognizing the input/output pin names. If you click on a pin you will see its connectivity (virtual) to the other pins. In the die border you can see input and output pins (to connected to the pads) and if you click on one of them you can see its connection to the cell pins.

What you see is only an abstract view of each cell, in which only its sizing, its orientation and the position and connectivity of its pins which will be used during the routing phase for connecting the cells among them.

Exploring placed design In the second line menu click on **Design Browser**. A new window opens useful to browse your circuit. Click for eaxmple on **maprca**: all the gates included in the RCA block are highligthed. it is also useful to select the "amoeba view" instead of the physical view. Now play with other sub-blocks...

Pre Clock-Tree-Synthesis (CTS) optimization Before synthesizing the clock tree we can try to optimize our design to achieve the required timing constraints. In the Main window select **Optimize** → **Optimize Design**, leave the default settings and click on OK.

Clock-Tree-Synthesis As the clock network could be very critical we perform a dedicated clock-tree synthesis so that a few buffers are used to reduce the load and the delay of clock tree. The clock synthesis process requires a file of constraints (.ctstch). This file can be provided by the user or generated automatically by the tool using the **.sdc** file produced by Design Compiler (see the first section of file design.globals). Thus, from top menu select **Clock** → **Synthesize Clock Tree**, a new window opens. Now we automatically generate the constraints for synthesize the clock tree. Click on **Gen Spec**..., a small window opens, choose CLKBUF_X1, CLKBUF_X2 and CLKBUF_X3 from the Cell List and click **Add** and then click **OK**. Now on the *Synthesize Clock Tree* window click **OK**. The buffers have been added. You can also browse from encounter the clock tree. From top menu select: **Clock** → **Browse Clock Tree**. Select **CLK** in the *Specified Clock List*, click select in the section named Clock Selection and **OK**. A new window opens in which you see the hierarchy of the clock tree buffers: select for example the first level buffer, then **Edit** → **Highlight**. The place is highlighted in the placed myfir view.

Post Clock-Tree-Synthesis (CTS) optimization Before running the routing we can try to optimize our design to achieve the required timing constraints. In the Main window select **Optimize** → **Optimize Design**. In the Design Stage section choose Post-CTS and in Optimization type check both **Setup** and **Hold**. Then, click **OK**.

Place filler. It is of help for technological reasons to complete the placement with filler cells. These will fill the holes to ensure continuity in N+ and P+ wells in each rows. From the top menu select **Place** → **Phyisical Cell** → **Add Filler**. A new window opens. Click Select and choose all the filler cells available and add them to the left list. These are possible fill cells among the placer will choose for filling the placement gaps. Now click OK and inspect what happens.

Routing. This phase is the last one: the connection among the cells will be performed using the available metal layers (the routing already visible is only a logical connection among cell pins, that have been used during the placement for taking into account the interconnection lenght). The routing steps is performed in two phases: the first is a sort of planning of the wire positions: **Route→TrialRoute**. The second is a detailed Routing in which violations and design rules (for example: a minimum distance between two metal wires must be respected so that during the fabrication a correct realization is assured) are carefully checked and solved: **Route→NanoRoute** to **Route** and OK to the Global and Detail Routing settings. Note in the encounter shell a few messages on the routing iteration steps for this optimization, together with some interesting data on the routing performed: the number of wires used for each layer, the total lenght routed in each layer, the number of via used. You still can analyze the metal layer used for the connections. The white boxes are violations which should be solved by a detailed operation that we will not perform.

Post routing optimization Note that at this point the design is complete (even if we dont have used real PADS all around the design). During this last step we can try to optimize our design to achieve the required timing constraints. Before starting the last optimization you have to issue on encounter command line (the shell where you launched encounter from) the command *setAnalysisMode -analysisType onChipVariation*. Then, in the Main window select **Optimize → Optimize Design**. In the **Design Stage** section choose **Post-Route** and in *Optimization* type check both **Setup** and **Hold**. Then, click **OK**.

Before going on, save your routed view: **Desing → Save Design** with default name the Encounter view. Hereinafter you will be able to import your design at the routed level. You can also save the design in DEF format or GDSII.

You can plot the image by the main menu: **Tools → Screen Capture → Screen Dump** and you can see the result using **Tools → Screem Capture → Display Screen Dump**. In your directory you have a new zipped file. Or (better) you can save from the same tool menu the GIF version.

We can now perform a few analyses for timing and integrity.

Parasitics For analyzing the time behavior Encounter uses the resistance and capacitance parasitic values for each metal wire. The extraction of such parasitics is the task of this step. The engine is able to compute the resistance and capacitance associated to each rectangle using its properties (technology and geometry information).

From the main manu select: **Timing → Configure MMMC**. Inspect the three possible RC Corners defined. We will use the standard.

From the main manu select: **Timing→ExtractRC** and select all the possible files to save. Moreover select standard as RC corner. Once done, look at the file SUM.spf (spf means stantard parasitics format): what does it rapresent? Note the *.subckt* instruction at the top file and the two "net section" (file top) and "instance section" (file bottom). And what about the SUM.spef file? Analyze the information contained in it and search the net with the greater capacitance and the greater lenght.
Read the file "SUM.setload" and "SUM.setres" which will be used later: what do they force?

Delay From the main manu select: **Timing→Report Timing** a new window opens. In *Design Stage* choose **Post-Route** and in *Analysis Type* choose **Setup**. Click **OK**. Repeat the same step choosing **Hold** in the Analysis Type section. In the directory named *timingReports* you find all the results produced by the timing analysis. Files .slk and .tarpt contain general and detailed informations on the timing paths and violations. Violations are referred to the timing constraints defined in the file **SUM.sdc**, that are identical to those used during the synthesis, are already loaded with the initial configuration file. Remember that slack means what is remaining between what you asked to have and what you actually have. So, if the slack is positive you are ok (you spared some time), if it is negative you are violating the constraint you decided. Remember that if you have not a real clock in the design you have to perform this step and specify a virtual clock.

Now complete timing analysis by selecting **Timing→ Debug timing**. A main window appears. There you have an overview of the critical paths that violated the constraint (i.e. the value respected at the synthesis level. If you click on one (the first for example) a new window appears that allow you to press element by element (in the horizontal bar) and see the corresponding delay and the path in evidence in the layout window. Play with these two windows in order to understand the potentials. it is also interesting to compare the delay found at synthesis level with the delay at the phyisical design level.... play and LEARN!!

Design analysis and verification Before ending the place and route phase we have to verify the connectivity and the design rules: **Verify → Verify Connectivity** and **OK**. Check the message produced by encounter and verify there are no violations. Usually violations are caused by

floating wires. To verify the design rules: **Verify** → **Verify Geometry** and **OK**. Check the message shown by encounter and verify there are no violations. Usually violations are caused by wrong constraints on the geometric feature during the place and route design flow. As an example if we require a narrow spacing between VDD and VSS layers when creating the power supply rings we may violate the design rules imposed by the technology we are employing. We can save area and gate count data as: **File** → **Report** → **Gate Count** and **OK**. Finally, we save i) the post place and route verilog netlist as **File** → **Save** → **Netlist** and **OK**; ii) the file with delay annotation (.sdf) as **Timing** → **Write SDF** and OK.

Further analyses YOU ARE LUCKY! (or maybe not...) due to problems with the new tool version and the new technology the electromigrationa nd IR drop analysis will be skipped. You can if you want play looking at the manual at the section: Rail Analysis.

Summary of what is requested

A dump of the layout, a saved placed and routed file with the encounter extension in the working directory, the extracted capacitance, resitance and spf file, the Delay report.

5.2 Physical design of your ADDER

You can play with the ADDER you optimized in previous lab. The exercise is to perform the previous steps (same choices) for the design and to analyze how the critical path changes.

You must generate file **ADDER.globals**: copy the SUM one, change the name to ADDER.globals and change the verlog file name. The references to all the libraries remains the same. You also should write the sdc file with the synthesis constraints.

Summary of what is requested

A dump of the layout, a saved placed and routed file in the working directory, the extracted capacitance, resitance and spf file, the Delay report .

5.3 Physical design of your MULTIPLIER

You can play with the MULTIPLIER you optimized in previous lab. The exercise is to perform the previous steps (same choices) for the design and to analyze the IR drop using a frequency of 2.5GHz and a switching activity of 0.4 for the power analysis . Now play and try to optimize the IR drop. You can increase the number of ideal VDD pad connections. You can start from the beginning and enlarge the power and ground rings, or increase and enlarge the power and ground stripes.
Of course this will be useful for the DLX project.

You must generate file **MULT.globals**: copy the SUM one and change the verlog file name. You also should write the sdc file with the synthesis constraints.

Summary of what is requested

A dump of the layout, a saved placed and routed file in the working directory, the extracted capacitance, resitance and spf file, the Delay report .