Slide 1

The aim of the lecture is to get an overview of place and route (P&R) steps.

Place and route follows the synthesis, which is explained in

http://ipe-iperic-srv1.ipe.kit.edu/doc/dds/ss18/lab/assignment2/Assignment2.html

and in the slides DDS19\_6\_Synthesis.pptx

Slide 2

The place and route procedure is performed by Cadence tool Innovus. It consists of several steps that are shown in the slide: design import, floorplan generation, power planning, special cell placing, placing, clock tree synthesis, routing and signoff.

The inputs for P&R are

1. The design as functional netlist (e.g. the Verilog netlist “.v” file)
2. The timing constraints file (“.scs” file – see explanation below)
3. The layout descriptions of the standard cells (pin positions, metal layer information) – so called “abstract” saved in the library exchange format “.lef” file
4. The electrical and layout descriptions of the standard cells (liberty “.lib” files – see below)
5. Information about metal layers, their capacitances, resistances, dimensions – qrc techfile.

The outputs of R&R are

1. The layout (graphical design station “.gds” file or open access (OA) data base)
2. The netlist for simulation (Verilog file) and the list of delays for every cell (standard delay format “.sdf” file)

The liberty files describe the ports (input and output, power), the type of cells: buffer, inverter, and gate, io pads, the operating condition, the power consumption and they give the timing modelling. These Liberty files are given by the foundry or could be generated for custom self-made standard cells with LIBERATE tool.

The sdc files (Synopsis Design Constraints) describe all the timing information between your design and the outside: What are the clocks signals? What are the relation between clocks? What is the delay between the signals and the clock? What is the input or output capacitance load? What are the timing exceptions?

Slide 3

Quite often chips contain analog and digital circuits. In this case, we are talking about a mixed mode chip.

There are two ways how a mixed mode chip can be designed. The first possibility is “analog on top”. The digital parts are automatically synthesized and layout is generated by the tools “Genus” and “Innovus”. As mentioned above, the outputs are the gds file (layout) and the Verilog file for simulations with annotated delays. The analog parts of the chip are designed by the tool “Virtuoso”. Virtuoso supports so called full custom design flow. The use of Virtuoso is the theme of the lecture “Design Analoger Schlatkreise”. It is possible to import the gds and the Verilog files generated by Innovus into Virtuoso and obtain the layout and the schematic view of the digital part. The top chip layout (that contains the analog- and the digital parts) is then drawn in Virtuoso by placing and connecting the blocks manually. Top-chip schematic is made as well. In the final design steps, the design rule check (DRC) and the layout versus schematics check (LVS) are performed. Timing between analog and digital blocks is not checked automatically.

Slide 4

In the case of “digital on top” design flow, analog blocks, which are designed in Virtuoso, are imported into Innovus and placed automatically. This design flow avoids manual routing of lines between analog and digital blocks. It checkes timing automatically. Analog blocks are taken by Innovus as an abstract view. Sdc file for the top module is needed. Lib file of the analog block could be provided as well.

Slide 5

P&R tool Innovus has graphical user interface (GUI) and allows performing of steps by mouse click and form filling. GUI can be used for learning, but it is impractical when whole P&R has to be repeated many times. Some commands are not accessible with the GUI.

Alternative to GUI use is performing of P&R from script. This is much better way when real design has to be done. There is often need to repeat the whole set of steps for many times. The script can be written in TCL tool command language. Scripts can be easily reused. Full set of commands is available.

The first set in P&R is importing of design. The netlist should be imported, top cell name specified, power and ground nets defined. Process node should be specified to tune the extraction of capacitances and resistances of metal lines.

Command examples

source XXX.globals

init\_design

setDesignMode -process 180

checkDesign –all

Slide 6

The next step in P&R is floorplanning. The floorplanning include following tasks: defining of design dimensions and area for input-output- (IO-) cells, pin placement, placing of reusable pieces of logic (hard-macros or intellectual property (IP) cells), creating of partitions in design, adding of well-tap cells. The output of the floorplan is the design exchange format “def” file. This file gives information about macros in design, placement, pin location, metal blockages, orientation. The def file can be also used as input for Genus tool, to perform a new iteration of synthesis. In this way, Syntesis will have idea about design size and optimize the netlist accordingly.

Slide 7

Choosing of correct size for floorplan is important. Floorplan should be big enough so that that there is space for all needed logic cells. The ratio of floorplan area and the space used for standard cells is called utilization. Too high utilization is also not good. It should be enough free space so that the clock buffers can be placed and that the tool has the freedom to move the cells and optimize timing. On the other hand, the floorplan should not be too big, because it may be too “empty” and the lines will be unnecessarily long which increases their capacitances and make circuits slow.

Slide 8

The maximum area for floorplan is also limited by production costs. Big chips are more expensive. Also, not entire chip area can be used by logic circuits. The input-output- (IO-) contacts (pads) take quite a lot of space. We distinguish here between the IO-bound designs (number of IO-pads is limiting factor) where the pads take the most of chip-space, because there are many input signals, are the core-bound designs where the core logic takes the most of the space.

In the following slides (9 - 30), we will discuss several topics related to floor planning: The choice of the chip area, IO pads, IP-blocks, hard-macros, well-tap cells…

Slide 9 and 10

To understand the production costs, let us introduce the three possibilities for chip production:

Full mask set run

Chip production is a repetitive process where different layers (n-well, p-well, active region, diffusions, polysilicon gate, metal layers, contacts, vias…) are produced on silicon wafers. The layers are structured using photolithography and for this photomasks are needed. In the case of full mask set run, one customer uses the entire mask set for his design or designs. For an 180nm process, typically 30 masks are needed. The size of the mask projection on the wafer, called the reticle, limits the maximum chip size. The maximum reticle size is typically 2.5cm x 2cm. There are typically 40 - 60 such reticles on one 200cm wafer. The full mask runs are good choice when the chips should be large and when many of them should be produced. The mask set is quite expensive but after making it, practically unlimited number of wafers can be produced. For an 180nm technology the mask set price is typically 80k€ and the wafer cost is 1.5k€.

Multi project wafer MPW “shuttle” run

In this case, many customers pay for a mask set and the reticle area is shared by many designs. The advantage is that the run cost is quite small for one customer. For an 180nm technology this may be 1k€ for mm2 of chip size. The disadvantage is that usually only several wafers are produced and that one customer obtains maximally several hundreds of chips. MPW runs are good for prototyping.

Multi layer mask (MLM) run

In this case, one mask is used for several layers. One customer pays for the mask set, and since less masks are needed than in the case of the full mask set run, the mask cost is reduced. When the masks are available, large number of wafers can be produced. However, the number of chips per wafer is smaller than in the case of the full mask set run and the maximum chip size is limited. This is illustrated in Slide 10.

Slide 11

IO-pads provide interface to external world. The IO-pad consists of the metal contact (the contact pad) and the IO-cell with electrostatic discharge (ESD) protection circuits and buffers. IO cells are in layout quite big. Like standard cells, they are designed to be abutted to each other.

Slide 12 and 13

The slide shows the analog pad. The ESD protection is usually done with two diodes that conduct current when the pad voltage exceeds the level vddp (typically 1.8V) or drops below the level vssp (typically 0V).

Slide 14

This slide shows the digital pad. Its structure is more complicated. It contains ESD protection and a level shifter, whose purpose is to translate the logic levels from (vss to vdd) to (vssio to vddio). The logic levels vss (logic zero) and vdd (logic one) are used by on-chip digital part (the core circuit). The levels vssio and vddio are used by the external circuits. In the case of the 65nm technologies vdd is usually 1.0V and vddio 1.8V. Vss = vssio = 0V. Level shifters are often bidirectional, they can transfer signals in both directions.

Slide 15

Power domains are used to separate various supplies. Typically we have analog and digital domains, but also multiple digital power domains are possible. Digital and analog domains are separated to avoid power supply cross talk. One digital domain can have a lower supply voltage to save power and another a higher voltage for better performances. Advanced options: Digital power domains can interface using level shifters placed by the tool. One digital power domain can be “shutdown” and put into a sleep mode.

Slide 16

IO cells have specific rules for power rail connections. When the cells are abutted the power rails will be shorted. In the case of IO cells that belong to different domains, spacing cells should be inserted to disconnect the rails.

Slide 17

The slide shows as an example two digital pads that belong to two power domains with the core voltages vdd1 and vdd2. The vdd rail is cut.

Slide 18 - 21

The slide explains the IO-cell and pad placement types.

IO cells can be placed outside the core area. In this case, we are talking about the ring IOs.

IO cells can be also placed inside the core area, this type of cells are called area IOs.

Connection pads can be placed around the die, which is typical for pads for wire bonding.

Connection pads can be also distributed all over the chip area for flip-chip connection.

Slide 22

IO cell types

IO Cells can be of various types, depending on the technology library.

Standard types:

Power connections: VDD, VSS, VDDIO, VSSIO

Fillers: FILLER1, FILLER10, CORNER

Signal cells for digital/analog signals

Special cells: differential signal (LVDS), oscillator cells, etc…

Slide 23

Partitions are used to cut-off the design in smaller subset. Each subset is synthesized and implemented separately.

Slide 24 and 25

Hard macros or IP blocks are design blocks similar to partitions, but provided by companies and ready to use. The hard macros are described by lef files that provide size and contact information. Most common hard macros are SRAMs and PLLs. They are placed like any other standard cell.

Slide 26

The slide shows the NMOS transistor. Notice the bulk contact, it is used to connect the p-type substrate (or p-well) with the negative supply voltage VSS or ground GND. Similarly, a PMOS transistor requires the n-well contact that is shorted with the positive supply voltage VDD.

Slide 27

If the p-well or n-well are not connected, their potentials can change and turn on parasitic bipolar transistors. This may lead to the “latch up”, an effect that causes a high current between VDD and VSS and may damage chip.

Slide 28 - 30

The p-well and n-well contacts can be placed in each standard cell as shown in slide 27 or they can be shared by several standard cells as shown in slide 28. In the latter case, the well contacts are part of the “well tap cell” that has to be placed separately. The well tap cell must not be too far from a MOSFET, otherwise the latch up may occur. To assure this the distance between well tap cells must not exceed typically 50µm.

Slide 31

The next step in P&R is power planning. Power connections are important, if they are too thin the voltage drop across the circuit can be large. This can affect the performances. Functional power analysis can be done but the best solution is to make power connection as good as possible and in this way avoid problems.

Slide 32 and 33

Power connections are made using command addStripe to add power stripes on high metal layers. Slide 33 shows the cross section. Also the power rings around core or blocks can be added.

Slide 34 and 35

Use the command editPowerVia to add VIA down to the power pins of the standard rows and macros. Consult the technology files (like lef) to find out on which layer the power pins are available. Cross section is shown in Slide 35.

Slide 36

Special cell placement.

Special cells like hard macros are placed using command placeInstance and by specifying coordinates.

Code example

placeInstance <instance\_name> <location> <orientation>

Slide 37

Placement of spare cells.

Some clusters of cells can be added on the core area for latter fixes. For instance we can add some spare inverters. The idea is the following: One set of wafers with chips would be produced completely and one set only up to metal one layer that is used by standard cells. The complexly produced chips can be then tested. If we discover a mistake that can be solved by connecting one of the spare cell (for instance one signal should be inverted), we can just modify the last metal layers produce new masks and accomplish the second set of wafers. The cost of such fixed production is much smaller than if we had to repeat all production steps.

Slide 38

Standard cell placement is quite trivial, the tool automatically place the cells where can be. The command is placeDesign. First routing is done to give a good idea of the feasibility of the design. The tool tries to optimize timing. This preliminary routing is then deleted. After placing the design is optimized using command optDesign –preCTS.

Code example

//Specify the top and bottom layer used for routing

setRouteMode -earlyGlobalMaxRouteLayer 4

setRouteMode -earlyGlobalMinRouteLayer 1

//Place your std cells

place\_opt\_design

//Optimize your design if necessary

optDesign -preCTS

Slide 39

Where are we?

Floor planning is done

Input-outputs are set

Power structures have been planned

Standard cells are placed

What remains?

Distribute the clock

Route the design

Respect design rules for manufacturing

Keep timing within acceptable margins

Export the design data for DRC/LVS and production

Slide 40

Before discussing clock tree synthesis let us remind the setup and hold times.

Setup time is the time available until the next clock edge, the input signal of a flip flop should change during this time.

Hold time is the time that is “not available” after a clock edge, the input signal of a flip flop should stay unchanged during this time.

Setup violation occurs when the data path is too slow.

Hold violation occurs when the data path is too fast.

There are several fixes for the violations, some of them can be done in design some of them are done by the P&R tool. For instance, the tool can add skew and start the next flip flop (the flip flop that receives signal) later by adding clock buffers to avoid setup violations.

Concerning hold violations, the tool can start the previous flip flop (the flip flop that generates signal) later or add delay buffers into the data path. This eliminates hold violations.

Notice that the hold problems cannot be fixed after the chip is produced. Setup violations can be avoided by slowing down the clock.

Slide 41

Optimization of timing.

The place and route is done based on timing data for the standard cells from the “liberty” .lib files and based on the constraints from the .scs file.

The liberty files describe the ports (input and output, power), the type of cells: buffer, inverter, and gate, io pads, the operating condition, the power consumption and they give the timing modelling. These Liberty files are given by the foundry or could be generated for custom IP with LIBERATE tool. The sdc files (Synopsis Design Constraints) describe all the timing information between your design and the outside: What are the clocks signals? What are the relation between clocks? What is the delay between the signals and the clock? What is the input or output capacitance load? What are the timing exceptions?

The timing of the cells depends on conditions such as supply voltage, temperature and on process variations. Different conditions are called corners. There is different library set for every corner. It can be fast, typical and slow corner for process variations. Each delay corner combines liberty (.lib) for the standard cells and the QRC tech files for the nets.

The chip may also have different functional modes (e.g. the “functional mode” with fast clock and the “test mode” with slow clock), that are described by sdc files for each mode.

It is possible to combine corners with modes in the P&R code, such combinations are called views. Further it is possible to tell the tool to use one specific view for the hold time analysis and another for the setup time analysis. For instance, it makes sense to use the view (functional mode + slow corner) for setup- and (functional mode + fast corner) for hold time optimizations.

Command examples

1

set\_analysis\_view

-setup [list name\_of\_view]

-hold [list name\_of\_view2]

2

create\_analysis\_view

-name name\_of\_view

-constraint\_mode name\_of\_mode

-delay\_corner name\_of\_corner

3

create\_constraint\_mode

-name name\_of\_mode

-sdc\_files path\_to\_sdc\_file

4

create\_delay\_corner

-name name\_of\_corner

-rc\_corner name\_of\_RCcorner

-early\_library\_set name\_of\_early\_lib\_set

-late\_library\_set name\_of\_late\_lib\_set

-early\_opcond\_library name\_of\_library

-late\_opcond\_libraryname\_of\_library

-early\_opcond name\_of\_cond

-late\_opcond name\_of\_cond

4

create\_rc\_corner

-name name\_of\_RCcorner

-T 25

-qx\_tech\_file path\_to\_qrcTechFile

5

create\_library\_set

-name name\_of\_lib\_set

-timing path\_to\_lib\_file

-si path\_to\_cdb\_file

Slide 42

In Cadence tools, the optDesign command is used to optimize timing after each phase: before clock tree synthesis, before routing, after routing.

Timing fixes are done for setup and hold separately, fixes may alter manufacturing rules (DRC), DRC may alter timing: always re-optimise the design along the implementation flow.

Code example

//Clock Tree Synthesis step (CTS) is done with CCOPT tool inside innovus

//Create a clock tree specification according to your sdc

create\_ccopt\_clock\_tree\_spec –filename ccopt.spec source ccopt.spec

//Run CCOpt

ccopt\_design –cts

//Report timing and optimize if necessary (slack <0)

timeDesign – postCTS

optDesign -postCTS

timeDesign – postCTS –hold

optDesign –postCTS –hold

//Report on clock trees

report\_ccopt\_clock\_trees  –filename clock\_trees.rpt

report\_ccopt\_skew\_groups –filename skew\_groups.rpt

Slide 43

Let us now discuss the clock tree synthesis. Clock is a special net, it has one clock source and many “sinks” – the clock receiving flip flops. Clock is in the design (in the Verilog code) just a net, however on chip it has usually a tree-like structure made of clock buffers. Clock tree is mandatory for synchronous design: one buffer cannot drive every flip flop. The clock should arrive almost at the same time at each flip flop. The aim of the clock tree synthesis is to produce the clock tree and assure that there are no setup and hold violations. This task is difficult. Clock net is very demanding: it toggles a lot which makes the large portion of power consumption. It toggles fast and it can influence other signals because of capacitive coupling: acts as signal Integrity aggressor. Since the lines are long it is difficult to drive and balance the clock tree.

Slide 44

Clock skew is the difference between the clock edges at two flip flops, one of them is generating the signal (the launch flip flop) and the other receiving it (the capture flip flop). Clock skew is defined as Skew = Tcapture – Tlaunch. It can be positive and negative. Positive skew leaves time for setup. Negative skew removes time for setup and it is good for hold. The original task of clock tree synthesis is to minimize the clock skew.

Slide 45

Clock latency is used to describe the time clock signal needs to reach the flip flops. It is measured from the clock source to the flip flop input.

Slide 46

There are two types of clock routing.

Standard clock tree is commonly used.

Clock mesh uses a grid structure to reduce the skew.

Clock mesh seems to be a better solution when there is a significant on chip variation of standard cell delays.

Slide 47

An advanced option of clock tree synthesis is to use positive/negative skew to improve setup or hold timing. This feature should be used with precaution.

Slide 48

Clock tree latency influences the capturing of input signals and generation of output signals. It can cause the input to register and register to output time violations. These violations are not serious as the setup and hold violations on chip because the external circuits can be modified to cope with too late outputs or they can generate the signals for the chip earlier or later.

Slide 49

The slide gives an example of timing report.

Slide 50

During routing process following is done:

Data paths are routed with accurate calculation (extraction) of line capacitances and resistances. During routing crosstalk and signal integrity are taken into account. Design rules are followed. Contacts between metal layers (the vias) are optimized by using double cuts. Antenna violations are fixed. The routing procedure can optimize the timing (timing driven routing) or try to avoid signal errors due to capacitive crosstalk (signal integrity driven routing). After routing connectivity and design rules are checked by commands verifyConnectivity and verifyGeometry. Despite these commands, design rule and layout versus schematic checks should be done. Timing is analysed and routing adapted if necessary with sets of commands timeDesign and optDesign.

Code example

//specify top and bottom layers for routing

setNanoRouteMode -routeTopRoutingLayer 4

setNanoRouteMode -routeBottomRoutingLayer 1

//turn on signal integrity driven routing

setsignoffOptMode -fixGlitch true | false

setNanoRouteMode -routeWithSiDriven true

//specify antenna fixing option

setNanoRouteMode -drouteFixAntenna true

setNanoRouteMode -routeAntennaCellName « my\_diode »

setNanoRouteMode -routeInsertAntennaDiode true

//route design

routeDesign

Slide 51 and 52

Antenna effect is a risk during manufacturing process. During production of routed lines, especially ion etching or polishing, charge is generated which increases the voltage of the lines. The high voltage can damage the transistor gates that are connected to the lines. IC would not be working if it happens. The problem can be fixed by adding electrostatic discharge protection diodes (antenna cells) or by making the metal bridges. A command verifyProcessAntenna will perform antenna violation fixing.

Slide 53

Some final steps should be done before manufacturing

Unused area is filled with filler cells with command addFiller. The cell names must be written in the code, they are found in technology documentation

Verify and fix DRC

Recalculate timing and fix setup and hold violations

Perform metal filling to fix density rules

Write out the GDS file

Perform layout versus schematic checks

Code example

//post route optimization flow

setExtractRCMode -engine postRoute

setExtractRCMode -effortLevel signoff

timeDesign -postRoute

optDesign -postRoute -setup

timeDesign -hold -postRoute

optDesign -postRoute -hold

addFiller -cell feedth feedth3 feedth9 -prefix FILLER

//do an eco route to fix DRC issue after Filler insertion

ecoRoute -fix\_drc

//signoff timing check

timeDesign –signoff

timeDesign –signoff –hold

report\_timing -machine\_readable > top.mtarpt

load\_timing\_debug\_report top.mtarpt

// Save the design in OA view

saveDesign -cellview {libname cellname viewname}

// Export the final netlist

saveNetlist main\_EDI.v

//export sdf for simulation

write\_sdf -view  <analysis view name>  my\_sdf\_file.sdf