This section intends to document the steps taken in the first stage of this dissertation. In order to understand the design flow used by Synopsys HARDIP team, a PLL design was made, covering all the stages of the backend flow. The major outcomes are shown below.

The VLSI design flow can be divided into two parts: Frontend design flow and Backend design flow. Both together, allow the creation of a functional chip from scratch to production. The frontend flow will be briefly described, while the backend flow is further analyzed.

**Frontend flow**

The frontend flow is responsible to determine a solution for a given problem or opportunity and transform it into a RTL circuit description. The stages of the frontend flow are identified in figure 1.

**Backend flow**

The backend process is responsible for the physical implementation of a circuit. It transforms the RTL circuit description into a physical design, composed by gates and its interconnections. The main phases of the backend process are Synthesis and Place&Route.

Figure 2 represents the flow of the backend process.

Each step is done by running Tcl (Tool Command Language) scripts that execute commands on the corresponding Synopsys tool.
Synthesis

Synthesis is responsible for converting the RTL description into a structural gate level based netlist. This netlist instantiates every element (standard cells and macros) that compose the circuit and its connections.

Synthesis can be described as follows:

Synthesis = Translation + Optimization + Mapping

Synopsys Design Compiler is the tool used to perform a logical synthesis. Its inputs are:

- The RTL description – Verilog or VHDL;
- The GTECH library – General technology library. Not tied to any specific technology (gates, flip flops);
- DesignWare Library –Synthetic library (adders, multipliers, comparators, etc).
- The standard cell library – the specific target library;
- The defined constraints – synthesis goals regarding timing, area, capacitance, max transition, fanout. Delivered by the Frontend team.
- Design Environment: The operating conditions (Libraries corners), wire load models.

First, DC reads the RTL description to its memory and translates it into an unmapped GTECH netlist. Then, considering the design constraints and design environment, DC compiles the GTECH netlist into the target library cells and optimizations are made to meet the design constraints. For this phase, all clock, sets and resets signals should be marked as ideal, since synthesis is a process with limitations regarding physical characteristics. Finally a set of reports are written and a gate level netlist is exported to be used by the place and route tool. Figure 3 shows the functional flow of synthesis using Synopsys Design Compiler.

**Figure 3 Synthesis**
Synthesis Verification

The first step is to verify a set of reports, which have information about timing, area, fanout and shows the violations to the defined constraints.

These reports must be interpreted to check if there are violations (setup time, hold times, area, max transition, etc.).

In case of violations DC can try to fix them by running optimization algorithms. If DC cannot fix the violations, one must go back toRTL coding. With these reports it is possible to check if the design is synthesizable and, therefore, if it is possible to proceed.

The final verification before proceeding to Place&Route is to run Formality, which is a logical verification tool. It takes the final netlist generated by DC and checks the logical equivalence with the RTL description.

Place&Route

Place&Route is the backend stage that converts the gate level netlist produced during synthesis into a physical design. Although the name denotes for two phases, the Place&Route stage can be divided in three steps: Placement, Clock Tree Synthesis (CTS) and Routing.

Placement involves placing all macros and cells into a certain and predefined space. It is done in two phases. The first one, called Coarse Placement, places the standard cells in order to optimize timing and/or congestion but not taking into account overlapping prevention. The second phase, which is named Legalize, eliminates overlap problems by placing the overlapping cells in the closest available space.

Clock tree synthesis is the creation of a balanced buffer tree in all high fanout clock nets to avoid violations regarding clock skew, max transition time, capacitance and setup and hold times.

Routing is responsible for designing all the wires needed to connect all cells of the circuit, while following the rules of the manufacture process. The connections between cells are done using metal layers placed one over the other and connected through vias. Routing has a negative impact on timing, transition and capacitance slacks. It introduces RC parasitic effects that cause delay, signal noise and increase IR drop. To minimize the parasitic impact, clock signals should be routed first and in middle metal layers, away from the noisy power supplies of the standard cells.

Routing is done in three phases: Global Routing (design routing nets), Track Assignment (assign nets to specific metal layers), and Search&Repair (fix violations).

Using Synopsys IC Compiler the design is, first, placed, followed by the clock tree synthesis (CTS) and, finally the routing of every cell. The result is a post-layout netlist and a GDS II file.
The steps taken to perform Place&Route using ICC are as follows:

**Placement**

1 - Create one empty milkyway library.

This library is linked to the technology file (physical rules) and the standard cells library (cells size, shape and pin location).

```bash
create_mw_lib ${design}_lib -technology $tech_file -open -mw_reference_library $mw_ref_lib
```

2 - Load Synthesis netlist

It is the netlist created by Design Compiler. It will be linked to the previously loaded physical and standard cells library.

```bash
read_verilog $verilog_file
```

3 – Connect the standard cells power pins to the design power supplies.

```bash
pg_connect $pwr_net $gnd_net $pwr_pin $gnd_pin
```

3 – Load TLU+ files

TLU+ files are provided by the foundry and give important information about the parasitic effects between cells and nets. This information will be used to correctly place and route all cells.

```bash
load_tlup $tlup_cworst $tlup_cbest $tlup_map
```

4 – Load the floorplan

The floorplan is the initial physical shape of the circuit. It has information about the circuit boundaries, the I/O pin location, the places where standard cells cannot be placed and the upper metal power straps. These straps are done in upper metal in order to have less resistance and smaller IR drop. The floorplan is previously done using Synopsys Custom Designer that produces the files to be loaded by ICC.

```bash
source boundary.tcl
source floorplan.tcl
source pins.tcl
source place_blk.tcl
pg_routing_by_metal
```

5 – Load the design constraints

Placement and routing are done in order not to violate these constraints.

```bash
set mode <functional_mode_name>
remove_sdc
```
source ${design}_constraints.tcl
set_library

6 – Check for special design constraints

Some standard cells libraries demand the use of special cells:

- Tap cells – Polarization cell that are added by rows.
- End cap cells – Placed for nwell continuity. Added in several ways like the beginning or end of each row.

7 – Antenna Issues

Antenna issues appear with the existence of huge single metal nets that accumulate a big amount of charge and can damage the connected transistors. As a solution diodes can be placed in all input and output ports to discharge to VSS. As an alternative a metal bridge can be added in an upper metal.

insert_diodes $diode_cell

8 – Load ICC internal settings

This is done to improve the timing correlation between ICC and PrimeTime, enable ICC to check for antenna issues during routing and to load other constraints.

source settings.place.tcl

9 – Perform the coarse placement

Place the cells inside the floorplan. They will be placed in order to meet timing but not avoiding overlapping cells.

create_placement –effort high –timing_driven –congestion –congestion_effort high

10 – Legalize the placement

Legalize is the process of eliminating overlap problems by placing overlap cells in the closest available space.

legalize_placement –effort high –incremental -timing

11 – Reorder the scan chain

In designs including scan chains, this step replaces the Q to SI (Serial Input) connections from too far way flip flops to closer ones.

read_def ${design}.scandef
optimize_dft
check_scan_chain
report_scan_chain
12 – Place and optimize spare cells

A set of spare cells (often called dummies) were added to the netlist in the final stages of synthesis and are, now, spread by the design to be used in future metal only ECO (Engineering Change Orders). Its inputs are connected to tie high or low cells (static 1 or 0).

spread_spare $num_spare
optimize_spare
set_attribute [all_spare_cells] is_fixed true
source dont_touch.tcl

13 – Connect the placed cells to power and ground

Connections are done by creating lower metal supply rails and connect them to the existing upper metal straps using vias.

pg_connect $pwr_net $gnd_net $pwr_pin $gnd_pin
set_preroute_drc_strategy –min_layer $min_layer –max_layer $max_layer
set via_size 2.5
set via_offset 1
pg_preroute $via_size $via_offset

14 – Create routing guides

Routing guides are used to avoid wrong routing pin access.

create_pin_route_guide $max_layer

Clock tree synthesis

15 – Define clock

Clock definition is done on the cell or port just behind the flip flop cloud.

remove_sdc
create_cts_clocks “<FF clock pin name>”
set_library

16 – Load clock tree synthesis settings

In order to improve CTS results, it is possible to define some setting like the maximum transition time, capacitance, the buffer level in the clock tree or double space between vias.
source settings.cts.tcl

17 – Remove all automatic added buffers
remove_clock_tree --clock_trees [get_clocks]

18 – Compile clock tree
compile_clock_tree --sync_phase_both --clock_trees [get_clocks]

19 – Check the clock tree for errors.
Check if any cell has big transition times, capacitance, or high fanout. Weak cells with high fanout produce huge transition times. In order to balance the clock tree it is possible to replace the weak cell by a stronger and logical equivalent one. On the other hand cells driving long nets can produce high transition times as well. One solution can be placing a strong buffer in the output of the driver cell.

20 – Set the clock don’t touch attribute
This is done to tell ICC to not touch these high sensitive nets in future setup and hold times optimizations.
mark_clock_tree --clock_synthesized
mark_clock_tree --clock_net
set_dont_touch_clk

21 - Perform a IR drop analysis
At this stage, before routing, it is important to check the IR drop over the circuit. ICC will output a color map representing the IR drop from the power straps and across all the circuit. The red spots on this map show high IR drops. It is also possible to have an idea of the circuit power consumption.
dump_ir_drop $pwr_net $gnd_net

22 – Run a Static Timing analysis from ICC
Running a STA from ICC at this stage allows detecting timing violations at this stage of the design. ICC can try to eliminate these violations automatically by accelerating or delaying paths.
load_sdc "<operation_condition>"

Automatic Optimization: psynopt

Routing

23 – Produce congestion map
In order to confirm if the design is routable, a congestion map should be produced. If a congestion problem arises, it will be reported.

route_zrt_global --congestion_map_only_true > route_congestion.rpt

24 – Route the clock nets

Route the high sensitive clock nets first.

route_zrt_group --all_clock_nets

25 – Route signal nets

At this stage ICC will try to preserve the previously routed clock nets.

route_zrt_auto --max_detail_route_iterations 20

26 – Check for errors

Check for DRC and timing errors. If big violations exist, ICC can try to fix them automatically.

route_opt --effort high --incremental

Setup time violations can be fixed by accelerating the path, which can be done by replacing cells by stronger and logical equivalent ones. To fix hold time violations the path has to be delayed. This is achieved by inserting buffers into the logical path.

27 – Place FILL and DCAP cells in the empty gaps

Place FILL DCAP cells to establish nwell continuity.

remove_stdcell_filler --stdcell

insert_stdcell_filler --avoid_layers $min_layer --cell_with_metal $decap_cells

insert_stdcell_filler --cell_without_metal $filler_cells

28 – Double vias for redundancy

This will improve silicon yield. Not all vias can be doubled because it has influence in timing and can cause DRC errors.

insert_zrt_redundant_vias --effort high --timing_preserve_setup_slack_threshold 0.2

-timing_preserve_hold_slack_threshold 0.2

29 – Run DRC and LVS

To check if the Place&Route process respected all rules.

verify_drc

verify_lvs
30 – Save the Milkyway database and the post-layout netlist

31 – Proceed to sign off

At this stage the post layout netlist is ready to be verified by the sign of timing, power, DRC and LVS tools.

Parasitic extraction

Parasitic extraction has the objective to create an accurate RC model of the circuit so that future simulations and timing, power and IR Drop analyses can emulate the real circuit response. Only with this information, all the analyses and simulations can report results close to the real functioning of the circuit. This way this stage needs to precede all signoff analyses.

Star RCXT is the Synopsys tool capable of performing parasitic extraction. It takes the post-layout Milkyway database and the NXTGRD files provided by the foundry (cells parasitic information) and produces SPEF (Standard Parasitic Exchange Format) and SBPF (Synopsys Binary Parasitic Format) files.

Static Timing Analysis (STA)

STA is a method to obtain accurate timing information without the need to simulate the circuit. It allows detecting setup and hold times violations, as well as skew and slow paths that limit the operation frequency.

Synopsys PrimeTime allows running STA over a physical design, for each corner. Taking as inputs the post-layout netlist and parasitic and standard cells information it outputs a series of reports, which give the possibility to detect timing violations. As mentioned before these timing violations can be fixed by inserting buffers or resizing cells. With PrimeTime it is possible to identify where to perform these modifications and test them. When a list of new buffers and resized cells is available, the modifications need to be done back in ICC, followed by another parasitic extraction and STA to check the results. This process is done iteratively until no violations are reported.

Post-layout Verification

Once again, formality should be run to check the logical equivalence of the post-layout netlist with the RTL description.

The huge number of transistors in a circuit can make the voltage level drop below a defined margin that ensures that the circuit works properly. IR Drop analysis allows checking the power grid to ensure that it is strong enough to hold that minimum voltage level. Synopsys PrimeRail is the tool that outputs IR-drop and EM analyses reports.
Then, PrimeTime-PX is the tool responsible of performing power analyses to estimate the power consumption of the circuit, for each corner. It has the capability of computing the dynamic and static power consumption of the whole design or the power consumption of each cell or macro.

The final step is to run Synopsys Hercules which is a DRC/LVS verification tool. DRC (Design Rules Checking) checks if the foundry geometric and connectivity rules are met. Examples of DRC’s include: Metal to metal spacing; well to well spacing; minimum metal width; Antenna Effect; Metal fill density. LVS (Layout Versus Schematic) checks if the physical circuit corresponds to the original circuit schematic.