**ASIC Synthesis**

Static 10-to-1 (1-to-10) CMOS SerDes blocks are synthesized using the same technique that was previously described for PCS block synthesis. In particular, RTL-level designs are synthesized into a gate-level Verilog netlist by means of Synopsys Design Compiler (SDC) and a commercial technology process library. The synthesis infrastructure, including a specially developed **flow\_defines.tcl**, **flow\_synth.tcl** and **flow\_write\_spice.tcl tcl** script files are located in the head catalogue of PMA/cmos\_synth\_and\_sim/CASE10\_1/synthesis/.

**flow\_defines.tcl** file provides linking to the definitions, logical descriptions and timing information of various logical gates (standard cells) of this particular library. It also defines all the global variables and design options that will be used during the synthesis. In general case, if any other library is used, these should be substituted with the corresponding cell library location information and environmental variables.

The translation of RTL-level design into a gate-level semantic is performed via invocation of commands specified in **flow\_synth.tcl** understandable to Synopsys Design Compiler. Each of synthesizable constructs is optimized against a specific design rules (defined partially in **flow\_synth.tcl**), optimization constrains (fanout, area and timing requirements for all design paths are specified in **flow\_defines.tcl** and **Sources/module\_name.sdc** Tcl scripts) and environmental descriptions (variations in voltage, temperature and modelling wire loads – all provided by **flow\_defines.tcl** and CMOS library itself).

The final result produced by Design Compiler is optimized on the gate-level and mapped to the library cell descriptions of the technology process in use (saved as Verilog HDL gate-level netlist).

Once the synthesis process has been completed **flow\_synth.tcl** script generates various report files providing: timing and area information, cells used in the design, clocks, ports and buses information, constraints, and other information. The obtained logs together with the synthesized designs are kept in **out.synth/** folder.

Timing closure analysis of synthesized designs is a major consideration that allows the circuit function properly. Synopsys DC performs static timing analysis checking the design paths for timing violations under worst conditions. There are common ways to rectify the setup and hold time violations inside the circuit, which are not described in this brief. The authors suggest using the obtained log files (**out.synth/**) as well as Design Vision graphical interface for design debugging and examination.

**flow\_write\_spice.tcl** script performs conversion of obtained Verilog gate-level netlist into HSPICE representation using corresponding standard CMOS HSPICE library source. In addition, HSPICE design instantiation file is generated which contains all I/O interfaces that can be mapped to testbench modules for design testing and verification.

**Post-synthesis HSPICE simulation**

Since any of the digital designs represented initially as a Verilog RTL code can be synthesized both to Verilog and HSPICE netlist descriptions, HSPICE environment presents one of the convenient ways for digital circuits integration with their analog counterparts. This section presents yet another way of digital design verification using HSPICE descriptions and HSPICE-compatible environments.

Once the digital design has been synthesized (usually kept in CASE10\_1/synthesis/out.synth/ folder), an HSPICE-compatible testbench infrastructure should be used for design verification.

CASE10\_1/simulation/ folder contains a “Digital Signal Injection” tool that automatically generates Spice stimulus (as well as Verilog testbench files) using specific input file (\*.dsi) format. Please refer to the DSI tool documentation and other examples supplied with the CASE10\_1/ SerDes design (test.dsi).

In order to run simulation, please undergo the following steps first:

1. Copy synthesized HSPICE designs as well as instantiation files from **out.synth/** folder into **hspice.source/** folder inside **CASE10\_1/synthesis/** catalogue.
2. Modify **P10110.instantiation.spi** in such a way that all modules are interconnected to each other, have correct input stimuli allocated and outputs are loaded.
3. In **CASE10\_1/simulation** folder create input stimuli file using the template provided – **test.dsi**. The names inside the input stimuli file should correspond to the names used inside **P10110.instantiation.spi** instantiation file. The produced Piecewise Linear Source (PWL) HSPICE input signals are kept in **tempdir.stimulus/** folder and generated each time the HSPICE simulation runs. The corresponding Perl script file – **flow\_dsi\_v0p7.pl** is used for this purpose.
4. Once input stimuli is created, please update the main HSPICE file - test.spi, which defines:
5. HSPICE technology process libraries used;
6. HSPICE simulation options; output signals that need to be probed; location of all dependencies, including input stimulus and files measuring the power footprint (POWER\_\*\*.spi files are also kept inside **CASE10\_1/synthesis/hspice.source/** folder );
7. Finally, all power supplies should be defined and assigned (instantiation file) for every circuit inside the system. These will not only provide a wider circuit debugging possibilities but also allow measuring the power profile of every circuit under different input conditions and loads.
8. Please create symbolic links of all files used in **synthesis/hspice.source/** folders into the main **simulation/** folder.
9. Run HSPICE simulation using **run\_hspice.sh** script file.

Once the simulation is completed the resulting waveforms can be analysed using either Synopsys CosmosScope or an open-source Gwave waveform viewers. Simulation outputs can be also reviewed using **hspice.out** output file.