Cadence On-Line Document

1 Purpose: Use Cadence On-Line Document to look up command/syntax in SoC Encounter.

2 Cadence On-Line Document

An on-line searching system which can be used to inquire about LEF/DEF Syntax and command line instructions, etc

2.1 Open the on-line document

% /home/raid1_1/linux/cadence/SOC/cur/tools/bin/cdnshelp &

2.2 Look-up LEF/DEF Syntax

Products → Languages → LEF/DEF 5.7 Language Reference → LEF Syntax

2.3 Look-up command line instructions

Products → SoC Encounter → …

Using "search" can help us to quickly find required commands (search topic, search manual, …). E.g., search "addstripe" (a command for power planning).

Source: These Labs are in the CIC standard flow (modified by Meng-Kai Hsu, 2011.11).

SOCE Lab (1/2): Circuit Placement & Power Planning

1 Purpose: Design import, Floorplan, Powerplan (Power Ring, Power Stripe)

2 Lab materials are available at "~cvsd/CUR/SOCE/SOCE_Lab.tar.gz".
Please install "MobaXterm" for graphical interface before lab (available from internet or "~cvsd/CUR/SOCE/MobaXterm_v2.2.zip").

3 We can first use vi, joe, or cat to see the details of CHIP.v, CHIP.ioc, and CHIP.sdc files in the design_data folder. CHIP.v is the main gate-level design. In additional to the original gate-level netlist after synthesis, a topmost module "CHIP" is added to include input/output pads and the original netlist (In this lab, a DCT circuit is used); CHIP.ioc is used to plan the physical positions of input/output/power/corner pads; CHIP.sdc is used to define timing constraints for timing-driven placement and routing.

3.1 Core power pad: provide power for the main core

3.2 I/O power pad: provide power for input/output pads

3.3 Corner pad: connect pad and pad power

4 Open SoC Encounter

4.1 % source /usr/cadence/CIC/soc.csh

4.2 % encounter

Note: DO NOT add "&" after the command "encounter" (background mode). We will use two different interfaces for SoC Encounter, (1) terminal (command line) mode, and (2) graphic user interface (GUI).

5 Design Import

5.1 Design → Import Design…

5.2 Verilog Netlist → Files → press the button …, and add the design file: "design_data/CHIP.v".

5.3 Verilog Netlist → Top Cell → fill in "CHIP"

5.4 Timing Libraries

Import following libraries as for step 5.2

Max Timing Libraries → fill in

library/lib/RF2SH64x16_slow_syn.lib
library/lib/slow.lib
library/lib/tpz013g3wc.lib

Min Timing Libraries → fill in

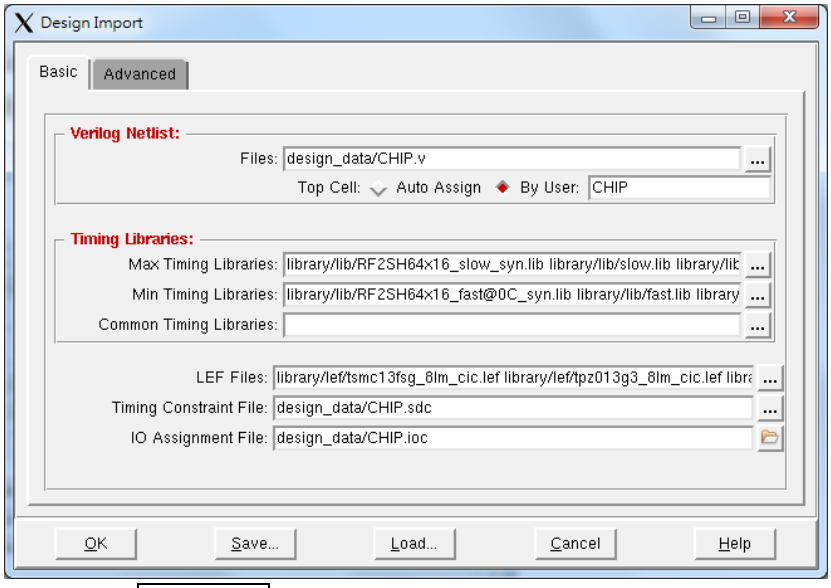
library/lib/RF2SH64x16_fast@0C_syn.lib
library/lib/fast.lib
library/lib/tpz013g3lt.lib

Max timing libraries are used to calculate steup time, and min timing libraries is used to calculate hold time. Besides, these libraries can also let SoC Encounter know which are inverters or buffers during timing optimization.

5.5 LEF Files → fill in
library/lef/tsmc13fsg_8lm_cic.lef
library/lef/tpz013g3_8lm_cic.lef
library/lef/RF2SH64x16.vclef
library/lef/antenna_8.lef
Note: Since the file: tsmc13fsg_8lm_cic.lef, includes all the process information, we should put tsmc13fsg_8lm_cic.lef on the first.

5.6 Timing Constraint File → fill in "design_data/CHIP.sdc"

5.7 IO Assignment File → fill in "design_data/CHIP.ioc"



5.8 Change to Advanced

5.9 Power

5.9.1 Power Nets → fill in "VDD"; Ground Nets → fill in "VSS" (TSMC process)

5.10 RC Extraction

5.10.1 RC Extraction → Typical/Best/Worst Capacitance Table File → fill in "library/tsmc013.capTbl"

5.10.2 QX → Tech File → fill in "library/tsmc13_8lm.cl/icecaps_8lm.tch"

5.10.3 QX → Library Directory → fill in "library/tsmc13_8lm.cl"

5.11 SI Analysis → CeltIC Libraries

5.11.1 Max cdB File → fill in "library/celtic/slow.cdB"

5.11.2 Min cdB File → fill in "library/celtic/fast.cdB"

5.11.3 Common cdB File → fill in "library/celtic/typical.cdB"

5.12 Since we need to enter a lot of settings to import designs, we can save settings after fill out the configuration by using the Save… button such that we can quickly restore the settings in the future (Load…).

5.13 We have kept a pre-saved profile, CHIP.conf, which can be directly loaded.

5.14 OK

6 Global Net Connect

In this step, we would like to connect the power/ground pins of all standard cells to VDD/VSS.

6.1 Floorplan → Connect Global Nets

6.1.1 Pin Name(s) → fill in "VDD" → To Global Net → fill in "VDD" → Add to List

6.1.2 Pin Name(s) → fill in "VSS" → To Global Net → fill in "VSS" → Add to List

6.1.3 Connect 1'b1/1'b0 to VDD/VSS.

6.1.3.1 Select Tie High → To Global Net → fill in "VDD" → Add to List

6.1.3.2 Select Tie Low → To Global Net → fill in "VSS" → Add to List

6.1.4 Apply → Check → Close

Note: In addition to connect 1'b1/1'b0 to VDD/VSS, we can also insert Tie high/Tie low cells (skip step 5.1.3); however, chip utilization might increase. Besides, if 1'b1/1'b0 do not connect to VDD/VSS, there will be some warnings after Check. In this moment, these warning can be ignored. (How to insert Tie high/Tie low cells will be explained later.)

7 Specify Scan Chain

Since we have inserted scan chain in the design, we need to specify where the scan chain is (scan in, scan out). We specify the scan chain by the command line mode:

7.1 encounter> specifyScanChain scan1 –start ipad_SCAN_IN/C –stop opad_SCAN_OUT/I

7.2 encounter> scantrace

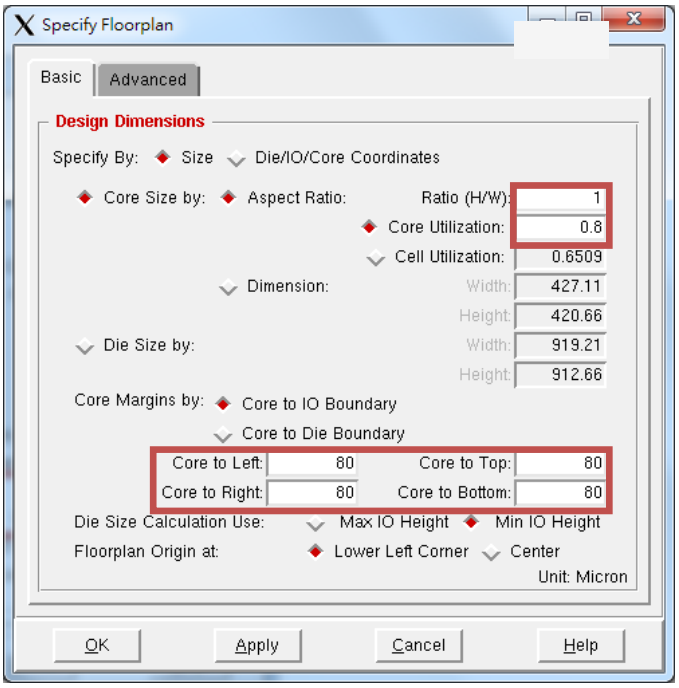
Note: In this example, the scan out is one of the primary outputs. In general, we need to specify ALL scan chains.

Q1: How many bits are there in the scan train (see the scan trace summary)?

_____

8 Floorplan

8.1 Floorplan → Specify Floorplan … (We should use appropriate settings for different designs.)



8.2 As we can see, the pink rectangle is the main design (DCT), and the green rectangle is the memory block (SRAM_i0).

8.3 We then roughly place the circuit into the core region by floorplanning.

Place → Standard Cells …

8.3.1 Select Run Placement In Floorplan Mode (deselect Include Pre-Place Optimization and Include In-Place Optimization in Optimization Options) → OK

8.4 After floorplanning, we can use different design views (Amoeba view, …) to see the results. In fact, we can arbitrarily change the orientation/position of the memory block. For example:

8.4.1 Select the memory block → Floorplan → Edit Floorplan → Flip/Rotate Instances… → select R90 → OK

We can find that the memory will be rotated by 90 degrees.

8.4.2 Change to the floorplan view → Select  → Move the memory (E.g., move the memory to upper right corner)

8.5 Add halos for hard blocks

In this step, we would like to add halos around hard blocks to avoid standard cells being placed near these hard blocks such that we can reserve more spaces for routing around these hard blocks.

8.5.1 Memory → Floorplan → Edit Floorplan → Edit Halo…

8.5.2 In the edit halo form, select Selected Blocks/Pads (make sure that the memory is selected) → Placement Halo → Top/Bottom/Left/Right → fill in “30um” → OK (There will be a red region around the memory.)

8.6 Timing analysis

8.6.1 Timing → Analysis Timing …

8.6.2 Select Pre-CTS in Design Stage → OK

The tool will start to do trial route and RC extraction, calculate circuit delay, and then use STA (Static timing analysis) for data paths.

8.6.3 We can see the analyzed result in the terminal. We should note that negative WNS (Worst Negative Slack) means the current placement result cannot satisfied the timing constraints in CHIP.sdc.

Q2: WNS is?_____; TNS is?_____.

8.6.4 In the WNS is negative, we can use timing optimization to improve the WNS. (Timing optimization will be explained later.)

8.7 After running placement in floorplan mode, we then run full placement.

8.7.1 Place → Standard Cells And Blocks …

8.7.2 Select Run Full Placement, deselect Include Pre-Place Optimization and select Include In-Place Optimization

8.7.3 Mode → Select Enable Clock Gating Cell Awareness → OK

8.7.4 OK

8.8 Place → Refine Placement to refine the cell orientations

8.9 Run timing analysis (step 8.6).

8.10 If the WNS is negative, we need to run timing optimization.

8.10.1 Timing → Optimize…

8.10.2 Use the default values and then press OK.

9 Save current files

9.1 Design → Save Design as → SoCE …

9.2 Use placed.enc for file name → Save

10 Create Power-ring

Power-rings are added around the core to avoid IR drop in the design.

10.1 Place → Refine Placement … → OK

This step is use to remove the trial route result after timing analysis.

10.2 Power → Power Planning → Add Rings …

10.2.1 Net(s) → fill in “VDD VSS”

10.2.2 Ring Configuration

10.2.2.1 Top/Bottom Layer → chose “METAL7 H”

10.2.2.2 Left/Right Layer → chose “METAL6 V”

10.2.2.3 Width → fill in “2”

10.2.2.4 Update

10.2.3 Change to Advanced

10.2.3.1 Select Use wire group → Interleaving

10.2.3.2 Number of bits → fill in "15"

10.2.3.3 OK

11 Connect power pads

11.1 Route → Special Route …

11.1.1 Net(s) → fill in "VDD VSS"

11.1.2 Select Pad pins in Route, and deselect others

12 Create Power-stripe

Power-stripes are added in the core to avoid IR drop in the design.

12.1 Power → Power Planning → Add Stripes …

12.1.1 Net(s) → fill in "VDD VSS"

12.1.2 Layer → select "METAL6"

Since we would like to create vertical stripe, METAL6 is used. If we want to create horizontal stripe, METAL7 might be used.

12.1.3 Width → fill in "1" → Update

12.1.4 Set Set-to-set distance to 100

12.1.5 Set X from left to 150 and X from right to 100

12.1.6 Change to Advanced

12.1.7 Select Use wire group and Interleaving, set Number of bits to 5

12.1.8 Select

12.1.8.1 Omit stripes inside block rings

12.1.8.2 Switch layer over obstructions

12.1.8.3 Pad/Core ring connection → Allow jogging

12.1.8.4 Block ring connection → Allow jogging

12.1.9 Change to Via Generation, and select

12.1.9.1 Use exact overlap area on partially intersection wires

12.1.9.2 Spilt vias while encountering Obs and different net Wires/Pins

12.1.9.3 Generate same-sized stack vias while encountering macro Pins/Obs

12.1.10 OK

Q3: How many groups of (group) power stripes are inserted? _____

13 DRC Check

13.1 Verify → Verify Geometry …

13.2 Check if there is any DRC error ("X" on the layout). We should clear violations as early as possible.

14 Save the files

14.1 Design → Save Design as → SoCE …

14.2 File name: powerplan.enc → Save