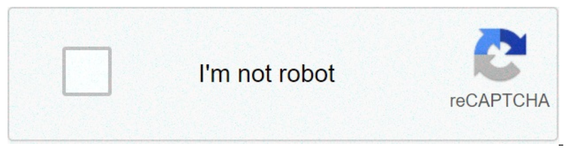

Continue

Synopsys design compiler manual

Synopsys design compiler reference manual. Synopsys design compiler constraints manual.

Synopsys Design Compiler A short tutorial: Connect @ -Aviral Mittal Ã¨ Download here the commad script for this tutorial Downlaod vhdl source code for this tutorial counter_top.vhd here counter.vhd here Download the resulting netlist on tsmc lib here Step 0. Evoke the compiler of design Ã¨ Ã Installare una libreria tecnologica. To synthesize a project, you need a technology library that will contain a description of the cells of the factory, and their timing. This is usually a .db file located in the installation directory of the library. To do this 1(a). Tell synopsys where your .db file is located. set search_path {/homes/amittal/s5/work/physical_lib/corelib/tsmc_090_g_art} Ã Ã 1(b). Tell synopsys what your technology library is, that you want to map your design over called set target_library {scadv_tsmc_cln90g_lvt_ss_0p9v_125c.db} Ã Ã 1(c). Set up the link libraries. These are optional .db files, which are pre-synthesized and ready to be read in EÃ 1 (d). Set up link libraries. These are optional .db files, which are pre-synthesized and ready to be read in set link_library {PLL10CCMID_W_125_1.35.db} Step 2. Read In your project files it is 2 (a). if verilog: if verilog is: if read_verilog counter.v is 2 (b). if vhdl: Come in questo tutorial read_vhdl counter_top.vhd is 2 (b). if vhdl: Come in questo tutorial read_vhdl counter_top.vhd is 2 (c). if ddc: read_ddc counter.ddc Step 3. Set Design Constraints: Ã¨ Ã¨ Ã¨ 3 (a) Set the Operating Frequency: You need to create a clock in the project, Ã¨ Ã Ã¨ With a given period of time. The command below creates a clock and calls it Ã âclk' time frame of the project. create_clock -periodo 10 -name design_clk (b) Set input constraints: Set the time taken by the signals that are coming into your project, outside of your project compared to the clock_input_delay 4.0 set_input_delay 4.0 [remove_from_collection [all_inputs] clk] -clock design_clk Ã© Ã© Ã© Ã Ã© 3 (c) Set output constraints: Set the time taken by the signals to leave your desing, outside of your project, before they are captured by Ã© Ã© Ã© It is the same orologio set_output_delay 7.0 [all_outputs] -clock design_clk It is the same orologio set_output_delay 7.0 [all_outputs] -clock design_clk That's it's instructing the design compiler using as little area as possible. set_max_area 0 Step 4. Enable gating clock for low power (optional) Ã 5. Write formal verification setupfile (optional) set_svf -append «counter.svf» Step 6: set the operating conditions set_operating_conditions Ã, Ã, Ã, Ã, ã, -max_libreria tcbn65lpwc_ccs Ã, Ã, Ã, Ã, Ã, Ã,-max wccom, Ã, Ã, Ã, ã, Ã, ã, -min_libreria tcbn65lpwc_ccs , Ã, Ã, Ã, Ã, Ã, Ã, Ã, Ã, ã, -min wccom step 7: set time derating. This serves to compensate for the variation of temporal paths on real silicon. set_timing_derate -min -late, 1.05 set_timing_date -min -early 1.00 step 8: Set the uncertainty of the clock, this is done to compensate for the skew waiting of the watch. set_clock_uncertainty -setup 0.500 [all_clocks] set_clock_uncertainty -holdÃ 0.000 [all_clocks] Step 9: Set the driving cell on set_driving_cellÃ input -lib_cell DFD1Ã -mult 1.0 -library tcbn65lpwc_ccs -pin Q [remove_from_collection [all_inputs] [collection_to_list [get_clocks]]]] Step 10: Set the load to all SET_LOG OUTPUT [EXPR 2 * [LOAD_OF [GET_PIN TCBN65LPWC_CCS / INVD4 / i]]]] [ALL_OUTPUTS] Step 11: If necessary Set Not Use on some Library cells SET_DONT_USE [GET_LIB_CELLÃ, TCBN65LPHVTWC_CCS / * *] Setting 12: Set the maximum trasition limit set_max_transition 0.500 [current_design] Step 13: Group some paths: It is always a good idea to group some paths to improve the optimization of the Default Synopsys routes works on the paths Worst. In the absence of groups, he will work on the worst street, which can be the desired one. The grouping of the paths will force the project compiler to work individually on the worst paths of each group. set ports_clock_root [get_ports [ALL_FANOUT -FLAT -CLOCK_TREE -LEVEL 0]] Group_Path -Name INP_PATHS -FROM [REMOVE_FROM_COLLECTION [ALL_INPUTS] $ ports_clock_root] -Root] -critical_root] Range 10.0 Step 14. VERIABLES Register Optimization set (optional) Ã , Ã, Ã, (a) set the automatic removal of constant flipflop (s) set compile_seqmap_propagate_constants true Ã, Ã, Ã, Ã, ã (b) set the automatic removal of flipflop downloaded Ã, Ã, Ã, ã, Ã , Ã, Ã, Ã, Ã, Ã, Ã, Ã, Ã, Ã, Ã, Ã, ã, Ã, set compile_delete_unloaded_sequential_cells false step 15. Set the mapping of synchronization reset to AVIOD XS in sims (optional) set hdlin_ff_alks_sync_set_reset Ã «TrueÃ» Step 16. Set up The higher level name as a current project and fill out the project Ã, Ã, Ã, Ã, (a) ã, current_design counter_top compile -map_efffort high Ã, Ã, Ã, Ã, Ã, (b) if you use DC Ultra: compile_ultra ã, Ã , Ã, Ã, Off Output Inversion of sequential cells compile_ultra -no_seq_output_inversion Step 17. Write the output in DDC write format -format ddc -output counter.ddc -hier Ã, Ã, ã, ã, 17 (b). Scrivere the € ™ Output in Verilog Write -format DDC -output Counter.VLog -Hier Step 18. You may want to flatten your project before writing NetList Ungroup -All -FlaTTTTTTEN WRITE -FORMAT Verilog -Output COUNTER_FLAT.VLOG Step 19. Write a report Timing of the Report_Timing Project> Counter_Timing.Rep Step 20. Exit the project quit Ã PiÃ¹ casuale shell DC ModalitÃ Tcl Comandi: define_design_lib lib1 -path ~/misc/vhdl analyze -library lib1 -format vhdl /homes/amittal/misc/vhdl/xx.vhdl get_design_lib_path SYNTH get_design__percorso lavoro read_verilog mse.v Ã report_timing -delay max -da -da ♪ I tried to find delays to an exit door, without limits, to form a point known in design. I get it. Then I wanted to find the next worst route to that exit door, to set a false path on the path found above. But it wouldn't work, so I created a CCCOK and I won the exit door,!! ♪ Report_Cell [ALL_REGISTERS] foulach_in_collection el [all_register] {Ads [get_object_name $ el]} set_handle [Apri my_file A] foulach_in_collection $ el [all_registers] {set reg_name [get_object_name $ el] puts $ file_handle _username and password dipartment ee. Make sure you are in your home directory. go to your directory sinsysys ofCD ~ Cadimedata / Synopsys This refers to the alias you set while you follow the configuration tutorial. Now you should be in the Usr / Local / Cadenceusers / Ðÿusername> / Synopsys directory. If you are not, type the following: CD Usr / Local / Cadenceusers / Synopsys takes a design at behavioral level Verilog and a default logical gates and produces a Gate Netlist Verilog. Synopsys requires a library file that describes the logical ports available for synthetic operations. Copy the library file to the Synopsys directory by typing: CP USR / LOCAL / CADENCEUSERS / FILES / LIBRARY.LIB. This file contains logical descriptions and timing information for a set of logical ports (cells). The first time you use Synopsys, you will probably not have a cell library, so you should use this library. The library file contains definitions for an inverter, Nand2, NAND3, NAND4, NOR2, NOR3, AOI12, AOI22, oai12, oai22 and d-flip-flop. When creating a cell library at Cadence later, you need to change the Synopsys library file to match the cells. If the cells differ from those in the library file, you may need to change the cell names, the pin names and / or logical descriptions in the library file to match; This should not be much of a problem if you pay attention to the Synopsys library file when you set cells in Cadence. Information synchronization is a more advanced topic, so you don't have to change the values indicated, even if you can want to experiment to change the delays reported to match those of the cells if the weather allows. After making changes to the library file (if necessary,) you need to compile it in a database format suitable for Synopsys to use. Start the LC_SHELL compiler by typing: LC_SHELL Now read and fill out the library file by typing: read_lib library. lib write_lib library -format db -output library. # Now you should see a compiled version of the library. LIB in the Synopsys Directory as Library.db. Open the sinopsys_dc.setup file with your favorite text editor and make sure the target_library line looks as follows: target_library = {library.db}; Save and close the file. Now copy your design behavioral Verilog in the Synopsys directory. It can help you to remove the trial module at this time: you can move it to a separate file to use later. You are ready to run Synopsys Design Vision. 1.2 Sinossis Design Vision Vision Vision Open Design Vision: To run Design Vision, type the following to start Design Vision: Design_Vision -DCSH_MODE & AMP; The Ampersand ('&') performs the process in the background, allowing you to enter UNIX commands while using Synopsys. Read files to synthesize: once the Synopsys window appears Vision, open your design by selecting File->Read from the menu bar. In the Read File dialog, select the Verilog file, select the format as verificalog and click OK. Close or minimize the report window that opens. An icon for your design should now appear in the main Select the drawing by clicking on it. Then insert it into it hierarchically by clicking the arrow down on the left toolbar or double-clicking the design icon itself. Create a symbol view: by pressing the "Create symbol view" button On the toolbar As shown below, you can generate a symbol view of the FSM with all the input and output pins specified. Fill out the design: Now you are ready to map the design. Select Design-> Fill Design from the menu bar and click OK in the opening window. The report window will provide a transcript of the mapping session as synopses converts your behavioral design into a gate-level netlist. Schematic view: when the mapping session is completed, select the schematic view icon from the toolbar to view the full gate level scheme. Report on the cell: to get an estimate of the number of cells needed to create the state machine, click "Design-> Cellsching Rapport". A window will appear the pop-up describing the list of cells to be used in the state machine. Generation of the mapped Verilog file: to create a Verilog gate level netlist for your design, select "File-> Save as". Do not specify the file name as original Verilog file with the addition of "_map.v" (IE: original: FSM.V ... Output: FSM_MAP.V) and select "Verilog" as file format from the drop-down menu. Click OK.Â Save drawings: It is important to save your work, since synopses may not remind you When he performs. Select File-> Save from the menu bar. By clicking OK in the Save Save As dialog box, Synos saves your job as a database format file that takes the name from the input paint file with the ".db" extension. Now Close Synopsys Design Vision by selecting File-> Exit. 1.3 • Preparation and simulation of the netlist of veralog The resulting Verilog file is a design gate-level netlist. It describes the circuit as a network of devices with names and pins as specified by the Synopsys Library. Â The netlist is not yet complete, however, since we must also add the descriptions of the modules for each of the devices. If you have used the Synopsys library file without making any changes, you must use the following file without modification. Otherwise, you may need to apply the changes as you did with the library file. The following file is a header that must be used with Verilog Netlist and contains descriptions of the module for logical gates that synossysys built the netlist with. Copy the file in your directory of Digitando synops: cp /usr/local/cadenceusers/files/header.v. Depending on the software used, the process ofof the header file with the mapped Verilog file varies as follows. Verilogger Users – To add this to the Verilog netlist, concatenate the two files (header.v First) together or edit the netlist and copy-paste the header at the beginning.Ã¨ You may want to differentiate the header netlist from the non-header netlist Editing Adding File "_map.v" to "_syn.v." Your gate-level Verilog netlist is now For verification. You should be able to simulate your verilog code mapped using the same testbench circuit used to simulate your behavioral code. You should be able to enable your test module with minor changes and check the functionality of the netlist with respect to the design of the behavioral level. Remember to remove the test module when importing your verification netlist or silicone ensembles. VCS users: do not concaten / copy-paste the header with the map.v. file Instead, as you start VCS, include the header file and MAP.V file in the command line. Ã, Ã, Ã, Ã, Ã, Ã, Ã, Ã, Ã, Ã, Ã, example: ã, vcs -ri -mupdate -line black_jack_fsm_map.vÃ, header.vÃ, black_jack_fsm_testrig.v by typing the above command , occurs if the NetList functions your port similar to the behavioral level of the FSM.ã, note: when simulating your structural code you need to initialize the values of all the inputs at the time equivalent to 0. 0.