Design Compiler Synthesis Tcl File

The following is a generic script that you can use to drive the synthesis process of *Design Compiler*. It can be run as-is for generic synthesis, or you can use it as a starting point for your own modified version.

```
# ***************************************************************
# * Author: Erik Brunvand, University of Utah
# *
# * General synthesis script for Synopsys. There should be
# * some general switches and parameters set in .synopsys_dc.setup
# * but other design-specific things are set here.
# * You should look carefully at everything above the
# * "below here shouldn't need to be changed" line.
# *
# * Note that lists that continue across a line need a backslash
# * to continue to the next line (if you have a bunch of
# * different verilog files, one per line, for example, or a bunch
# * of target library files). Make SURE there isn't a space after
# * the \ because that can cause Synopsys to complain...
# *
# * Once you've modified things to your project, invoke with:
# *
# * syn-dc -f syn-script
# *
#
#  This script assumes that the following variables are defined
#  in the .synopsys_dc.setup file.
#
#  SynopsysInstall = path to synopsys installation directory
#  synthetic_library = designware files
#  symbol_library = logic symbols for making schematics
#
#
###################################################################
# below are parameters that you will want to set for each design
###################################################################
set myFiles [list <your-HDL-files>] ;# list of all HDL files in the design
set fileFormat verilog              ;# verilog or VHDL
set basename <module-name>          ;# Top-level module name
set myClk <clk>                     ;# The name of your clock
set virtual 0                       ;# 1 if virtual clock, 0 if real clock

# Timing and loading information
set myPeriod_ns <num>               ;# desired clock period (in ns) (sets speed goal)
set myInDelay_ns <num>              ;# delay from clock to inputs valid
set myOutDelay_ns <num>             ;# delay from clock to output valid
set myInputBuf <cellname>           ;# name of cell driving the inputs
set myLoadLibrary <libname>         ;# name of library the cell comes from
set myLoadPin <pinname>             ;# name of pin that the outputs drive

# Control the writing of result files
set runname <string>                ;# Name appended to output files
```

```
# the following control which output files you want. They
# should be set to 1 if you want the file, 0 if not
set write_v 1              ;# compiled structural Verilog file
set write_ddc 1            ;# compiled file in ddc format (XG-mode)
set write_sdf 0            ;# sdf file for back-annotated timing sim
set write_sdc 1            ;# sdc constraint file for place and route
set write_rep 1            ;# report file from compilation
set write_pow 0            ;# report file for power estimate

# compiler switches...
set useUltra 1                          ;# 1 for compile_ultra, 0 for compile
                                         # mapEffort, useUngroup are for
                                         # non-ultra compile...
set mapEffort1 medium                   ;# First pass - low, medium, or high
set mapEffort2 medium                   ;# second pass - low, medium, or high
set useUngroup 1                        ;# 0 if no flatten, 1 if flatten
##########################################################
# Set some system-level things...
#
# Your library path may be empty if your library will be in
# your synthesis directory because "." is already on the path.
set search_path [list . <your-library-directory>\
[format "%s%s"  SynopsysInstall /libraries/syn] \
[format "%s%s"  SynopsysInstall /dw/sim_ver] \
    ]

# Target library list should include all target .db files
# Library names separated by spaces if more than one
set target_library [list <your-target-lib>.db]
# synthetic_library is set in .synopsys_dc.setup to be
# the dw_foundation library.
set link_library [concat  [concat  "*" $target_library] $synthetic_library]

#*********************************************************
#*   below here shouldn't need to be changed...          *
#*********************************************************

# analyze and elaborate the files
analyze -format $fileFormat -lib WORK $myFiles
elaborate $basename -lib WORK -update
current_design $basename

# The link command makes sure that all the required design
# parts are linked together.
# The uniquify command makes unique copies of replicated
# modules.
link
uniquify

# now you can create clocks for the design
# and set other constraints
if {  $virtual == 0 } {
   create_clock -period $myPeriod_ns $myClk
} else {
   create_clock -period $myPeriod_ns -name $myClk
}


# Set the driving cell for all inputs except the clock
# The clock has infinite drive by default. This is usually
# what you want for synthesis because you will use other
# tools (like SOC Encounter) to build the clock tree
# (or define it by hand).
if {  $virtual == 0 } {
```

```
      set_driving_cell  -library $myLoadLibrary -lib_cell $myInputBuf [all_inputs] \
   } else {
      set_driving_cell  -library $myLoadLibrary -lib_cell $myInputBuf  \
           [remove_from_collection [all_inputs] $myClk]
   }

   # set the input and output delay relative to myClk
   if {  $virtual == 0 } {
      set_input_delay $myInDelay_ns -clock $myClk [all_inputs] \
   } else {
      set_input_delay $myInDelay_ns -clock $myClk \
       [remove_from_collection [all_inputs] $myClk]
   }
   set_output_delay $myOutDelay_ns -clock $myClk [all_outputs]

   # set the load of the circuit outputs in terms of the load
   # of the next cell that they will drive, also try to fix
   # hold time issues
   set_load [load_of [format "%s%s%s%s%s" $myLoadLibrary "/" $myInputBuf "/" $myLoadPin]] [all_outputs]
   set_fix_hold $myClk

   # This command will fix the problem of having
   # assign statements left in your structural file.
   # But, it will insert pairs of inverters for feedthroughs!
   set_fix_multiple_port_nets -all -buffer_constants

   # now compile the design with given mapping effort
   # and do a second compile with incremental mapping
   # or use the compile_ultra meta-command
   if {  $useUltra == 1 } {
      compile_ultra
   } else {
      if {  $useUngroup == 1 } {
        compile -ungoup_all -map_effort $mapEffort1
        compile -incremental_mapping -map_effort $mapEffort2
     } else {
        compile -map_effort $mapEffort1
        compile -incremental_mapping -map_effort $mapEffort2
     }
   }

   #
   check_design
   report_constraint -all_violators

   #*************************************************************
   #* now write out the results                                 *
   #*************************************************************

   set filebase [format "%s%s" [format "%s%s" $basename "_"] $runname]

   # structural (synthesized) file as verilog
   if {  $write_v == 1 } {
      set filename [format "%s%s" $filebase ".v"]
      redirect change_names \
   { change_names -rules verilog -hierarchy -verbose }
      write -format verilog -hierarchy -output $filename
   }

   # write out the sdf file for back-annotated verilog sim
   # This file can be large!
   if {  $write_sdf == 1 } {
      set filename [format "%s%s" $filebase ".sdf"]
      write_sdf -version 1.0 $filename
```

```
}

# this is the timing constraints file generated from the
# conditions above - used in the place and route program
if {  $write_sdc == 1 } {
    set filename [format "%s%s" $filebase ".sdc"]
    write_sdc $filename
}

# synopsys database format in case you want to read this
# synthesized result back in to synopsys later in XG mode (ddc format)
if {  $write_ddc == 1 } {
    set filename [format "%s%s" $filebase ".ddc"]
    write -format ddc -hierarchy -o $filename
}

# report on the results from synthesis
# note that > makes a new file and >> appends to a file
if {  $write_rep == 1 } {
    set filename [format "%s%s" $filebase ".rep"]
    redirect $filename { report_timing }
    redirect -append $filename { report_area }
}

# report the power estimate from synthesis.
if {  $write_pow == 1 } {
    set filename [format "%s%s" $filebase ".pow"]
    redirect $filename { report_power }
}

quit
```