DC Synthesizer Lab 24 Handout

Here is an example of a synthesis script for the SerDes speed optimization part of the lab, assuming the TSMC tcbn90hgptc library. It is based to a great extent on the Deserializer constraints applied in Lab23.

```
#
#########################################################
# The initial commands here are appropriate
# for use of the TSMC synthesis libraries:
#
set hdlin_translate_off_skip_text "true"
set verilogout_no_tri  "true"
set default_schematic_options     "-size infinite"
set write_name_nets_same_as_ports "true"
#
# End TSMC commands.
#########################################################
#
# dc_shell TcL startup script:
#
set designer "verilogUserN"
set company  "SVTI"
#
# Some design environment variables:
#
set search_path ". [getenv SYNOPSYS]/../TSMCLibes/tcbn90ghpSYN"
#
# xxxtc = Typical; xxxbc = Best; xxxwc = Worst:
set target_library tcbn90ghptc.db
set link_library   tcbn90ghptc.db
#
set symbol_library tcbn90ghp.sdb
#
# --------------------------------
# Messages to suppress (VERY CAREFULLY!):
  # Parameter ranges are new:
  set NoMessageList "VER-311"
  # Integers are signed:
  set NoMessageList "$NoMessageList VER-314"
  #
suppress_message $NoMessageList
#
# --------------------------------
```

```
#
define_design_lib SerDes -path ./SerDesSynth
#
analyze -work SerDes -format verilog SerDes.v
analyze -work SerDes -format verilog   Serializer/Serializer.v
analyze -work SerDes -format verilog   Serializer/SerEncoder/SerEncoder.v
analyze -work SerDes -format verilog   Serializer/SerialTx/SerialTx.v
analyze -work SerDes -format verilog   Deserializer/Deserializer.v
analyze -work SerDes -format verilog   Deserializer/DesDecoder/DesDecoder.v
analyze -work SerDes -format verilog   Deserializer/SerialRx/SerialRx.v
analyze -work SerDes -format verilog     FIFO/FIFOTop.v
analyze -work SerDes -format verilog     FIFO/FIFOStateM.v
analyze -work SerDes -format verilog     FIFO/DPMem1kx32.v
analyze -work SerDes -format verilog     PLL/PLLTop.v
analyze -work SerDes -format verilog     PLL/ClockComparator.v
analyze -work SerDes -format verilog     PLL/MultiCounter.v
analyze -work SerDes -format verilog     PLL/VFO.v
#
elaborate -work SerDes SerDes
#
# --------------------------------
#
set_operating_conditions NCCOM
set_wire_load_model -name "TSMC64K_Lowk_Conservative" [all_designs]
#
# For XG mode portability to back-end tools:
set_fix_multiple_port_nets -all
#
set verilogout_higher_designs_first "true"
#
# --------------------------------
# Design Rules:
#
set_drive      10.0 [all_inputs]
set_load       20.0 [all_outputs]
set_max_fanout 30   [all_inputs]
set_max_fanout 20   [all_designs]
#
# --------------------------------
#
# module-specific constraints:
#
set_max_area  0
set_max_delay 8 -to [all_outputs]
#
```

```
#
# Additional timing constraints to uncomment (yours may be different):
#
# Squeeze for speed (scales uncertainty down):
# Actual requirement = 1000.
#set ParPeriod 400
#
#set Delta [expr $ParPeriod/1000.0];
#set RDelta [expr $Delta*8.0]
#set WDelta [expr $Delta*5.0]
#set PLLDelta $Delta
#
#set VFOClockIn  [get_pins Des_U1/DesDecoder_U1/ParClk]
#set VFOClockOut [get_pins Des_U1/SerialRx_U1/PLL_RxU1/VFO1/ClockOut]
#set ParOutClock [get_ports Des_U1/ParOutClk]
#set DecodeClockIn [get_pins Des_U1/DesDecoder_U1/SerClk]
#
#create_clock -period $ParPeriod $ParOutClock
#create_clock -period $ParPeriod $VFOClockIn
#create_generated_clock $VFOClockOut -source $VFOClockIn -multiply_by 32
#create_generated_clock $DecodeClockIn -source $VFOClockOut -multiply_by 1
#
#set_clock_uncertainty $RDelta   [get_clock $ParOutClock]
#set_clock_uncertainty $WDelta   [get_clock $VFOClockIn]
#set_clock_uncertainty $PLLDelta [get_clock $VFOClockOut]
#set_clock_uncertainty $PLLDelta [get_clock $DecodeClockIn]
#
#set_propagated_clock [all_clocks]
#set_fix_hold [all_clocks]
#
# Problem-signal constraints:
#set_max_delay 15 -from [all_inputs] -to [all_outputs]
#set_max_delay 10 -from [all_inputs] -to Des_U1/FIFO_U1/FIFO_Mem1/ParityErr
#set_max_delay 10 -from [all_inputs] -to Des_U1/FIFO_U1/FIFO_Mem1/Storage*/*
#set_max_delay  2 -from $DecodeClockIn -to Des_U1/DesDecoder_U1/*FrameSR*/Q
#
# To force fix of hold violations:
#set_cost_priority {min_delay max_transition max_delay max_capacitance max_fanout}
#
#
compile
#
write -hierarchy -format verilog -output SerDesNetlist.v
#
write_sdf SerDesNetlist.sdf
#
report_area > /dev/null
report_area
echo "Report area:"              > SerDesNetlist.log
report_area                     >> SerDesNetlist.log
report_timing
echo "Report timing:"           >> SerDesNetlist.log
report_timing                   >> SerDesNetlist.log
```

```
report_net_fanout -threshold 100
echo "Report net fanout:"        >> SerDesNetlist.log
report_net_fanout -threshold 100 >> SerDesNetlist.log
check_design
echo "Check design:"             >> SerDesNetlist.log
check_design                     >> SerDesNetlist.log
#
```

```
#
# ============================================================
# Here are some example problem solutions:   Suppose that
# the compile run found two high-fanout nets (>1000) named below.
#
# net = Des_U1/FIFO_U1/RegFile1/ClockW
# net = Ser_U1/FIFO_U1/RegFile1/ClockW
#
# The following buffering is for illustration, only.
# The nets being buffered are clock nets, and clock-tree
# buffer insertion is best done during floorplanning.
# Insertion of buffers in the logical netlist, as done
# here, may make buffering more difficult during the
# floorplanning!
#
#current_design DPMem1kx32_AdrHi4_DWid32_0
#set_drive 10 [all_inputs]
#balance_buffer -force -net [find net ClockW]
#
#current_design DPMem1kx32_AdrHi4_DWid32_1
#set_drive 10 [all_inputs]
#balance_buffer -force -net [find net ClockW]
#
#current_design SerDes
#report_net_fanout -threshold 100
##
# Recompile to ensure delays OK:
#compile -incremental_mapping
#report_net_fanout -threshold 100
#
#write -hierarchy -format verilog -output SerDesNetlist_Recompile.v
#
#report_area > /dev/null
#report_area
#echo ""                              >> SerDesNetlist.log
#echo " ========================== " >> SerDesNetlist.log
#echo "After incremental recompile:"  >> SerDesNetlist.log
#echo ""                              >> SerDesNetlist.log
#report_area
#echo "Report area:"                  >> SerDesNetlist.log
#report_area                          >> SerDesNetlist.log
#report_timing
#echo "Report timing:"                >> SerDesNetlist.log
#report_timing                        >> SerDesNetlist.log
#report_net_fanout -threshold 100
#echo "Report net fanout:"            >> SerDesNetlist.log
#report_net_fanout -threshold 100     >> SerDesNetlist.log
#check_design
#echo "Check design:"                 >> SerDesNetlist.log
#check_design                         >> SerDesNetlist.log
#
# Drop into interactive mode:
# (End)
```