
DO NOT SHARE
SLIDES AND CLASS MATERIALS
ON ONLINE SITES
Course Hero

CSEE W4823 Lab#3

Ashish Shukla

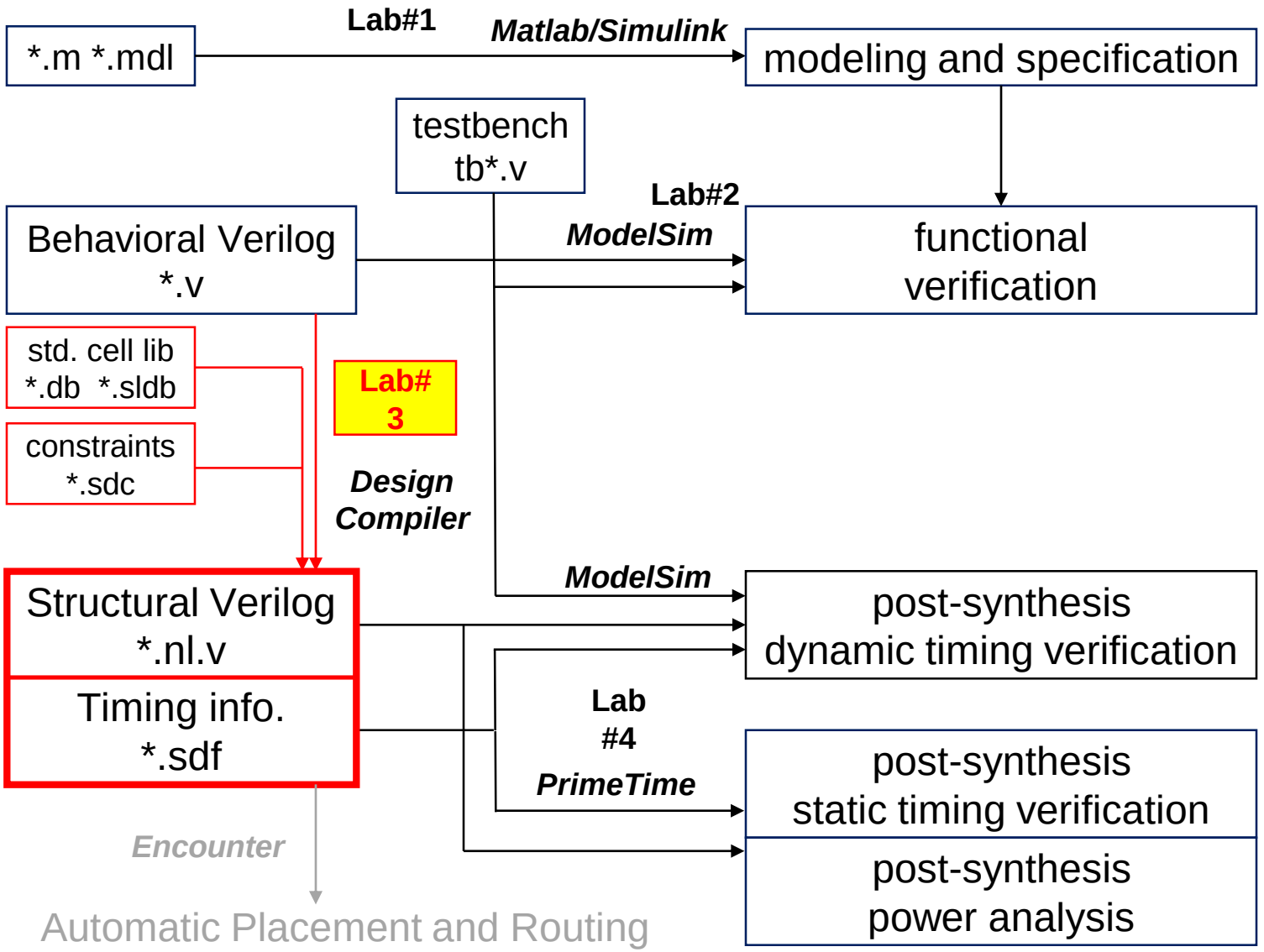
7th Oct, 2021

Acknowledgement: Mingoo Seok & Previous TAs, Columbia

Topics

- Lab#1: Design flow & Matlab®
- Lab#2: Verilog HDL / ModelSim®
- Lab#3: Synthesis / Design Compiler®
- Lab#4: Timing and power analysis / PrimeTime®
- Lab#5: Memory Compiler
- Lab#6 and following labs: Project based lab

Semi-Custom Flow



Overview of Design Compiler

- Translate *.v file (RTL code) to *.nl.v file (gate-level netlist)
- Inputs of Design Compiler
  - RTL code of your design (*.v) : lfsr1.v
  - Standard cell library (*.db, *.sldb) : common.tcl
  - Timing constraints (*.tcl) : timing.tcl
  - Synthesis flow commands/setup (*.tcl) : lfsr1.tcl
- Outputs of Design Compiler
  - Gate level netlist (*.nl.v)
  - Timing information (*.sdf)
  - Synthesis report (*.rpt)

4 Steps to Run Synthesis

- Read Design and Library
- Set Design Constraints
  - Timing constraints
  - Loading constraints
  - Other constraints, like max fanout
- Compile
- Generate Reports and Check

These 4 steps are organized in ***lfsr1.tcl***

Before you start

- Copy folder
  - copy /courses/ee6321/share/4823-fall2020/dc/ to your [...]/project/
  - MAINTAIN THE SAME FILE/FOLDER STRUCTURE!
- Take a quick view of those file
  - use “./nuke.sh” to delete all the generated reports
- DC tutorial is in /dc/doc/ folder

Step 1: Read Design and Library

- lfsr1.tcl (This file can be found in /dc/lfsr1/ folder)

```
##########################################
# READ Design and Library                #
##########################################
set top_level lfsr1
source -verbose "../common_script/common.tcl"
read_verilog {../../rtl/$top_level/$top_level.v}
set set_fix_multiple_port_nets "true"
list_designs

if { [check_error -v] == 1 } { exit 1 }
```

Library file: common.tcl

- This file can be found in /dc/common_script/ folder

```
set search_path [list "." ]
set synthetic_library [list "dw_foundation.sldb"]
set link_library [list "*" \
                       "/courses/ee6321/share/ibml3rflpvt/synopsys/scx3_cmos8rf_lpvt_tt_lp2v_25c.db" \
                       "dw_foundation.sldb"]
set target_library "/courses/ee6321/share/ibml3rflpvt/synopsys/scx3_cmos8rf_lpvt_tt_lp2v_25c.db"
```

standard cell library

- Search_path: The path to search for unresolved reference library or design
- Synthetic_library: DesignWare library
- link_library: The library that DC uses to resolve cell references.
- target_library: The technology that design map to.

- If you have your own standard cells *.db file (covered in EE6321) you can add here to tell design compiler to use them for mapping.

- You can also use set_dont_use CELLNAME to tell design compiler to not use certain cells when mapping.

Step 2: Set Design Constraints

- lfsr1.tcl (This file can be found in /dc/lfsr1/ folder)

```
##########################################
# Design Constraints                     #
##########################################
current_design $top_level
link
check_design
source -verbose "./timing.tcl"
set_max_capacitance 0.005 [all_inputs]
set_max_fanout 4 $top_level
set_max_fanout 4 [all_inputs]
set_max_area 0
set_fix_multiple_port_nets -all -buffer_constants
```

read timing constraints

max capacitance value for a net

set the max fanout that DC can use

set the target area "0" means optimize for smallest possible

- Design rule constraint: max_capacitance, max_fanout
- Optimization constraint: max_area, max_delay
- Tool will give priority to design rule constraints during optimization

Step2: Set timing Constraint File: timing.tcl

- This file can be found in /dc/lfsr1/ folder

```
# Setting variables
set clk_period 2.000
set clk_uncertainty 0
set clk_transition 0.010
set typical_input_delay 0.05
set typical_output_delay 0.05
set typical_wire_load 0.005

#Create real clock if clock port is found
if {[sizeof_collection [get_ports clk]] > 0} {
  set clk_name "clk"
  set clk_port "clk"
  #If no waveform is specified, 50% duty cycle is assumed
  create_clock -name $clk_name -period $clk_period [get_ports $clk_port]
  set_drive 0 [get_clocks $clk_name]
}

#Set clock uncertainty
set_clock_uncertainty $clk_uncertainty [get_clocks $clk_name]
#Propagated clock used for gated clocks only
set_clock_transition $clk_transition [get_clocks $clk_name]

# Configure the clock network
set_fix_hold [all_clocks]
set_dont_touch_network $clk_port
set_ideal_network $clk_port
#set_ideal_network pad_*
#set_ideal_network sc_*

# Set the paths to be ignored in timing opt
#set_false_path -from pad_*
#set_false_path -from sc_*

# Set input and output delays
set_driving_cell -lib_cell INVX1TS [all_inputs]
set_input_delay $typical_input_delay [all_inputs] -clock $clk_name
remove_input_delay -clock $clk_name [find port $clk_port]
set_output_delay $typical_output_delay [all_outputs] -clock $clk_name

# Customize for block
#set_output_delay 52 [all_outputs] -clock $clk_name
#set_output_delay 0 next_* -clock $clk_name

# Set loading of outputs
set_load 0.005 [all_outputs]
```

define variables

Find real clock
Or create virtual clock (vclk) if no clock is needed for timing reference

Set clock uncertainty/slew rate

Set_input_delay: amount of time the input signal is available after clock edge

Set_output_delay: amount of time the the signal is required before clock edge on output signals

Set load cap values for all the output ports

Step 3 & 4: Compile and Write Reports

- lfsr1.tcl

```
########################################
# Compile                              #
########################################
check_design
#uniquify
current_design $top_level
link
compile_ultra
```

check for errors/warnings

COMPILE!!

avoid using “tri” nets

```
########################################
# Write outputs                        #
########################################
source -verbose "../common_scripts/namingrules.tcl"
set verilogout_no_tri TRUE
write -hierarchy -format verilog -output "${top_level}.nl.v"
#write_sdf -context verilog "${top_level}.temp.sdf"
write_sdc "${top_level}.syn.sdc" -version 1.7
write_sdf "${top_level}.syn.sdf"
# Generate report file
set maxpaths 20
set rpt_file "${top_level}.dc.rpt"
check_design > $rpt_file
report_area  >> ${rpt_file}
report_power -hier -analysis_effort medium >> ${rpt_file}
report_design >> ${rpt_file}
report_cell >> ${rpt_file}
report_port -verbose >> ${rpt_file}
report_compile_options >> ${rpt_file}
report_constraint -all_violators -verbose >> ${rpt_file}
report_timing -path full -delay max -max_paths $maxpaths -nworst 100 >> ${rpt_file}
report_timing -delay max -nworst 1 -max_paths 10000 -path end -nosplit -unique -sort_by slack > ${top_level}.syn.critical_regs
report_timing -delay max -nworst 1 -max_paths 10000 -path full -nosplit -unique -sort_by slack > ${top_level}.syn.critical_regs.full
report_timing -delay max -nworst 1 -max_paths 10000 -path end -nosplit -unique -sort_by slack -to [all_outputs] > ${top_level}.syn.critical_regs.output
report_timing -delay max -nworst 1 -max_paths 10000 -path end -nosplit -unique -sort_by slack -to [all_registers -data_pins] > ${top_level}.syn.critical_regs.regs
report_timing -delay min -nworst 1 -max_paths 10000 -path short -nosplit -unique -sort_by slack > ${top_level}.syn.fast_path
quit
```

generate outputs
(1) gate level netlist: *.nl.v
(2) timing information: *.syn.sdf/sdc
(3) DC report: *.dc.rpt

clarify items that report after synthesis

Run the compiler

- in /dc/lfsr1/
  - type `sh synth.sh` to run the script
- After a while, the compile process is complete, let's look at the output files.
  - gate-level netlist (*.nl.v)
  - design constraints (*.syn.sdc)
  - timing information (*.syn.sdf)
  - DC-generated report (*.rpt)
    - area
    - timing
    - power

Output file: lfsr1.nl.v

```
////////////////////////////////////////////////////////////
// Created by: Synopsys DC Ultra(TM) in wire load mode
// Version   : O-2018.06-SP5-1
// Date      : Thu Oct  8 15:15:11 2020
////////////////////////////////////////////////////////////


module lfsr1 ( clk, resetn, seed, lfsr_out );
  input [15:0] seed;
  output [15:0] lfsr_out;
  input clk, resetn;
  wire   N3, N4, N5, N6, N7, N8, N9, N10, N11, N12, N13, N14, N15, N16, N17,
         N18, n50, n60, n70, n80, n90, n100, n110, n120, n130, n140, n150,
         n160, n170, n180;

  DFFQX1TS lfsr_out_reg_1_ ( .D(N4), .CK(clk), .Q(lfsr_out[1]) );
  DFFQX1TS lfsr_out_reg_2_ ( .D(N5), .CK(clk), .Q(lfsr_out[2]) );
  DFFQX1TS lfsr_out_reg_3_ ( .D(N6), .CK(clk), .Q(lfsr_out[3]) );
  DFFQX1TS lfsr_out_reg_4_ ( .D(N7), .CK(clk), .Q(lfsr_out[4]) );
  DFFQX1TS lfsr_out_reg_6_ ( .D(N9), .CK(clk), .Q(lfsr_out[6]) );
  DFFQX1TS lfsr_out_reg_7_ ( .D(N10), .CK(clk), .Q(lfsr_out[7]) );
  DFFQX1TS lfsr_out_reg_8_ ( .D(N11), .CK(clk), .Q(lfsr_out[8]) );
  DFFQX1TS lfsr_out_reg_9_ ( .D(N12), .CK(clk), .Q(lfsr_out[9]) );
  DFFQX1TS lfsr_out_reg_10_ ( .D(N13), .CK(clk), .Q(lfsr_out[10]) );
  DFFQX1TS lfsr_out_reg_11_ ( .D(N14), .CK(clk), .Q(lfsr_out[11]) );
  DFFQX1TS lfsr_out_reg_13_ ( .D(N16), .CK(clk), .Q(lfsr_out[13]) );
  DFFQX1TS lfsr_out_reg_14_ ( .D(N17), .CK(clk), .Q(lfsr_out[14]) );
  DFFQX1TS lfsr_out_reg_15_ ( .D(N18), .CK(clk), .Q(lfsr_out[15]) );
  DFFQX1TS lfsr_out_reg_12_ ( .D(N15), .CK(clk), .Q(lfsr_out[12]) );
  DFFHQX4TS lfsr_out_reg_0_ ( .D(N3), .CK(clk), .Q(lfsr_out[0]) );
  DFFQX1TS lfsr_out_reg_5_ ( .D(N8), .CK(clk), .Q(lfsr_out[5]) );
```

Output file: lfsr1.syn.sdc

```
#####################################################################

# Created by write_sdc on Thu Oct  8 15:15:11 2020

#####################################################################
set sdc_version 1.7

set_units -time ns -resistance kOhm -capacitance pF -voltage V -current mA
set_max_fanout 4 [current_design]
set_max_area 0
set_driving_cell -lib_cell INVX1TS [get_ports clk]
set_driving_cell -lib_cell INVX1TS [get_ports resetn]
set_driving_cell -lib_cell INVX1TS [get_ports {seed[15]}]
set_driving_cell -lib_cell INVX1TS [get_ports {seed[14]}]
set_driving_cell -lib_cell INVX1TS [get_ports {seed[13]}]
set_driving_cell -lib_cell INVX1TS [get_ports {seed[12]}]
set_driving_cell -lib_cell INVX1TS [get_ports {seed[11]}]
set_driving_cell -lib_cell INVX1TS [get_ports {seed[10]}]
set_driving_cell -lib_cell INVX1TS [get_ports {seed[9]}]
set_driving_cell -lib_cell INVX1TS [get_ports {seed[8]}]
set_driving_cell -lib_cell INVX1TS [get_ports {seed[7]}]
set_driving_cell -lib_cell INVX1TS [get_ports {seed[6]}]
set_driving_cell -lib_cell INVX1TS [get_ports {seed[5]}]
set_driving_cell -lib_cell INVX1TS [get_ports {seed[4]}]
set_driving_cell -lib_cell INVX1TS [get_ports {seed[3]}]
set_driving_cell -lib_cell INVX1TS [get_ports {seed[2]}]
set_driving_cell -lib_cell INVX1TS [get_ports {seed[1]}]
set_driving_cell -lib_cell INVX1TS [get_ports {seed[0]}]
set_load -pin_load 0.005 [get_ports {lfsr_out[15]}]
set_load -pin_load 0.005 [get_ports {lfsr_out[14]}]
set_load -pin_load 0.005 [get_ports {lfsr_out[13]}]
....
set_load -pin_load 0.005 [get_ports {lfsr_out[0]}]
```

- An elaborate version of your constraints in lfsr1.tcl

Output file: lfsr1.dc.rpt (area)

```
****************************************
Report : area
Design : lfsr1
Version: O-2018.06-SP5-1
Date   : Thu Oct  8 15:15:11 2020
****************************************

Library(s) Used:

    scx3_cmos8rf_lpvt_tt_1p2v_25c (File:
/courses/ee6321/share/ibm13rflpvt/synopsys/scx3_cmos8rf_lpvt_tt_1p2v_25c.db)

Number of ports:                           34
Number of nets:                            64
Number of cells:                           46
Number of combinational cells:             30
Number of sequential cells:                16
Number of macros/black boxes:               0
Number of buf/inv:                         11
Number of references:                       7

Combinational area:               246.240002
Buf/Inv area:                      50.400002
Noncombinational area:            408.959991
Macro/Black Box area:               0.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:                  655.199994
Total area:                 undefined
```

BE AWARE: This is just area estimation from synthesis

(Doesn’t have actual interconnect/floor plan)

Output file: lfsr1.dc.rpt (timing)

```
****************************************
Report : timing
        -path full
        -delay max
        -nworst 100
        -max_paths 20
Design : lfsr1
Version: O-2018.06-SP5-1
Date   : Thu Oct  8 15:15:11 2020
****************************************

Operating Conditions: tt_1p2v_25c   Library: scx3_cmos8rf_lpvt_tt_1p2v_25c
Wire Load Model Mode: top

  Startpoint: lfsr_out_reg_5_
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: lfsr_out_reg_0_
            (rising edge-triggered flip-flop clocked by clk)
  Path Group: clk
  Path Type: max

  Point                                    Incr       Path
  -----------------------------------------------------------
  clock clk (rise edge)                    0.00       0.00
  clock network delay (ideal)              0.00       0.00
  lfsr_out_reg_5_/CK (DFFQX1TS)            0.00       0.00 r
  lfsr_out_reg_5_/Q (DFFQX1TS)             0.74       0.74 f
  U50/Y (XOR2X1TS)                         0.26       1.00 r
  U51/Y (XOR2X1TS)                         0.28       1.28 f
  U52/Y (AO22X1TS)                         0.45       1.73 f
  lfsr_out_reg_0_/D (DFFHQX4TS)            0.00       1.73 f
  data arrival time                                   1.73

  clock clk (rise edge)                    2.00       2.00
  clock network delay (ideal)              0.00       2.00
  lfsr_out_reg_0_/CK (DFFHQX4TS)           0.00       2.00 r
  library setup time                      -0.26       1.74
  data required time                                  1.74
  -----------------------------------------------------------
  data required time                                  1.74
  data arrival time                                  -1.73
  -----------------------------------------------------------
  slack (MET)                                        0.01
```

Individual contribution to Path delay

Path delay

data required time = 1.74 = T_{CLK}-(DFF setup time)

1.73ns=actual path delay

Slack: Negative indicates constraints violation

make sure it MET!

Output file: lfsr1.dc.rpt (power)

```
****************************************
Report : power
        -hier
        -analysis_effort medium
Design : lfsr1
Version: O-2018.06-SP5-1
Date   : Thu Oct  8 15:15:11 2020
****************************************


Library(s) Used:

    scx3_cmos8rf_lpvt_tt_1p2v_25c (File: /courses/ee6321/share/ibm13rflpvt/synopsys/scx3_cmos8rf_lpvt_tt_1p2v_25c.db)


Operating Conditions: tt_1p2v_25c   Library: scx3_cmos8rf_lpvt_tt_1p2v_25c
Wire Load Model Mode: top


Global Operating Voltage = 1.2
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1pW


--------------------------------------------------------------------------------
                                   Switch   Int      Leak     Total
Hierarchy                          Power    Power    Power    Power    %
--------------------------------------------------------------------------------
lfsr1                              1.82e-02    0.258   844.583    0.277 100.0
```

This is just power estimation from synthesis for NOMINAL VDD (1.2V) There is no information about the inputs, so it may not be accurate

Ways to study

- If you do not understand what a command is doing…
  - dc_shell>man xxx (e.g. man set_input_delay)
  - check user manual
  - search on google
- Good luck!