Name – Manjunath Veeraiah Kalmath

USN - 01FE17BEC240

Roll No – 175

School of ECE

Q.1 Using the "ls -l" command followed by a grep, create a list of only links in your home directory (there may be only one: rb.v). Copy and paste the output in your submission document.

```
adlv30@VLSI18:~/Desktop
File  Edit  View  Search  Terminal  Help
[adlv30@VLSI18 Desktop]$ ls
adld  cadence_RAK_lib  NanGate_15nm_OCL  Nangate_OCL_lib  rb.v  RISCV(RV32IM)  skywater_lib
[adlv30@VLSI18 Desktop]$ ls -l | grep rb.v
lrwxrwxrwx. 1 adlv30 users   46 Nov  8  2020 rb.v -> /cad_area/vlsi/design/rb/rb_flat_synthesized.v
[adlv30@VLSI18 Desktop]$
```

Q.2 Verilog statements end in semicolons (;). Many of the Verilog statements in the netlist run over multiple lines. The line with a semicolon at the end represents last piece of the statement. Two consequent statements from a Verilog netlist are listed below to illustrate this point:

AOI22_X1 \u0_rb_offset/U6339 (.A1(\u0_rb_offset/n3119), .A2(\u0_rb_offset/n10641), .B1(\u0_rb_offset/cfg_reg [1229]), .B2(\u0_rb_offset/n10642), .ZN(\u0_rb_offset/n10645));
INV_X2 \u0_rb_offset/U1640 (.A(\u0_rb_offset/n10888), .ZN(\u0_rb_offset/n5975));

a. Modify the netlist so that there is no line break in any Verilog statement. Hint: you can replace the new line character ("\n") with a space, unless the line ends in a semicolon. Also, look out for "endmodule" statements. Save the modified netlist as "rb_flat_single_line.v". Save this file in a directory called "rb" in your home directory.

```
adlv30@VLSI18:/cad_area/vlsi/design/rb
File  Edit  View  Search  Terminal  Help
[adlv30@VLSI18 rb]$ ls
$                          flop_inst_clock_node_adlv30.txt   rb_flat_synthesized.v              sample1.txt
AOI22_ver.v                flop_inst_clock_node_adlv30.txt~  rb_flat.v                          sample2.txt
flop_inst_clock_node1.txt  flop_inst_clock_node.txt          rb_hierarchy_synthesized.v         sample3.txt
flop_inst_clock_node1.txt~ flop_inst_clock_node.txt~         rb_placed_netlist_07Jan1033.v      sample.txt
flop_inst_clock_node1.v    rb_flat_mode.v                    rb_post_cts_netlist_07Jan1033.v    SPAOI_X1_adlv30.
flop_inst_clock_node2.txt  rb_flat_mod.v.                    rb_post_route_netlist_07Jan1033.v  SPAOI_X1.v
flop_inst_clock_node2.txt~ rb_flat_single_line1.v            rb_scan_flat.v                     SPAOI_X2.v
flop_inst_clock_node3.txt  rb_flat_single_line_adlv30.v      rb_scan_hier.v                     test.txt
flop_inst_clock_node3.txt~ rb_flat_single_line.v             rb_synthesized_hier_old.v
[adlv30@VLSI18 rb]$
```

```
File  Edit  View  Search  Terminal  Help
// Verilog file generated from Magma Bedrock database// Thu Jan  6 16:49:50 2011// Bedrock Root: Magma_root// Entity:rb
:rb Library:workmodule rb (rb_ior_anlg_en_o, rb_ior_antestsel0_o, rb_ior_antestsel1_o,      rb_ior_bypass_o, rb_ior_cal
t_mode_o, rb_ior_cal_strn_o,      rb_ior_cal_strp_o, rb_ior_cg5_clk_su_sl_o, rb_ior_cg6_clk_su_sl_o,      rb_ior_cg7_cl
l_o, rb_ior_cg8_clk_su_sl_o, rb_ior_cg9_clk_su_sl_o,      rb_ior_cpll_antestsel_o, rb_ior_cpll_cpdac_o, rb_ior_cpll_ext
_o,      rb_ior_cpll_fbintdivsel_o, rb_ior_cpll_fbshdivsel_o, rb_ior_cpll_pdivsel_o,      rb_ior_cpll_posedgeen_o, rb_i
l_refclksel_o, rb_ior_cpll_refdivsel_o,      rb_ior_cpll_resetpllcpb_o, rb_ior_cpll_vcocfg_o, rb_ior_data_probe_en_o,
_ior_en_clktree_o, rb_ior_en_div_cclkout_o, rb_ior_en_mulph_div4_o,      rb_ior_en_syspll_fbkdiv2_o, rb_ior_fuse_slew_c
rb_ior_fuse_spare_o,      rb_ior_hpi_o, rb_ior_jtag_en_cg_o, rb_ior_jtag_en_misc_in_o,      rb_ior_jtag_en_misc_out_o,
_jtag_ocascreqclk2xa_o,      rb_ior_jtag_ocascreqclk2xan_o, rb_ior_jtag_ocascreqclk2xb_o,      rb_ior_jtag_ocascreqclk2
 rb_ior_jtag_ocascreqclk2xc_o,      rb_ior_jtag_ocascreqclk2xcn_o, rb_ior_jtag_ocascreqinst_o,      rb_ior_jtag_ocascre
al_o, rb_ior_jtag_ocascreqoper_o,      rb_ior_jtag_ocascreqparity_o, rb_ior_jtag_ocascreqparsedata_o,      rb_ior_jtag_
_o, rb_ior_jtag_onphaseerr_o, rb_ior_jtag_orplclk2xa_o,      rb_ior_jtag_orplclk2xan_o, rb_ior_jtag_orplclk2xb_o,
_jtag_orplclk2xbn_o, rb_ior_jtag_orplclka2_o,      rb_ior_jtag_orplclka2n_o, rb_ior_jtag_orplclkb2_o,      rb_ior_jtag_
kb2n_o, rb_ior_jtag_orpldataa_o, rb_ior_jtag_orpldatab_o,      rb_ior_jtag_orplparitya_o, rb_ior_jtag_orplparityb_o,
ior_jtag_orplvalida_o, rb_ior_jtag_orplvalidb_o, rb_ior_mode_bypass_o,      rb_ior_nc0_o, rb_ior_nc0_oe_n_o, rb_ior_nc1
_ior_nc1_oe_n_o,      rb_ior_nc2_o, rb_ior_nc2_oe_n_o, rb_ior_nc4_o, rb_ior_nc4_oe_n_o,      rb_ior_nc5_o, rb_ior_nc5_o
 rb_ior_ocascreqinst_slice0_o,      rb_ior_ocascreqinst_slice1_o, rb_ior_ocascreqinst_slice2_o,      rb_ior_ocascreqinst
e3_o, rb_ior_ocascreqinstval_slice0_o,      rb_ior_ocascreqinstval_slice1_o, rb_ior_ocascreqinstval_slice2_o,      rb_i
screqinstval_slice3_o, rb_ior_ocascreqoper_slice0_o,      rb_ior_ocascreqoper_slice1_o, rb_ior_ocascreqoper_slice2_o,
_ior_ocascreqoper_slice3_o, rb_ior_ocascreqparity_slice0_o,      rb_ior_ocascreqparity_slice1_o, rb_ior_ocascreqparity_
_o,      rb_ior_ocascreqparity_slice3_o, rb_ior_ocascreqparsedata_slice0_o,      rb_ior_ocascreqparsedata_slice1_o, rb_
ascreqparsedata_slice2_o,      rb_ior_ocascreqparsedata_slice3_o, rb_ior_odamod_o, rb_ior_onintr_35_o,      rb_ior_onpha
_o, rb_ior_orpldataa_slice0_o, rb_ior_orpldataa_slice1_o,      rb_ior_orpldataa_slice2_o, rb_ior_orpldataa_slice3_o,
ior_orpldatab_slice0_o, rb_ior_orpldatab_slice1_o,      rb_ior_orpldatab_slice2_o, rb_ior_orpldatab_slice3_o,      rb_i
lparitya_slice0_o, rb_ior_orplparitya_slice1_o,      rb_ior_orplparitya_slice2_o, rb_ior_orplparitya_slice3_o,      rb_
plparityb_slice0_o, rb_ior_orplparityb_slice1_o,      rb_ior_orplparityb_slice2_o, rb_ior_orplparityb_slice3_o,      rb_
rplvalida_slice0_o, rb_ior_orplvalida_slice1_o,      rb_ior_orplvalida_slice2_o, rb_ior_orplvalida_slice3_o,      rb_ior
validb_slice0_o, rb_ior_orplvalidb_slice1_o,      rb_ior_orplvalidb_slice2_o, rb_ior_orplvalidb_slice3_o,      rb_ior_o
en_cg_o, rb_ior_pad_dstrn_o, rb_ior_pad_dstrp_o,      rb_ior_pad_strn_o, rb_ior_pad_strp_o, rb_ior_prb_add_o,      rb_i
 detect_rst_o, rb_ior_rcv_lv0_hv1_o, rb_ior_rst_o,      rb_ior_scan_mode_o, rb_ior_sel_bsc_o, rb_ior_sel_latchup_o,
```

b. Using “grep” to get all the Verilog statements that instantiate flip-flops (flip-flop library cells have the string “DFF” in their names), generate a list of flip-flop instances and the name of the net or port connected to it’s clock pin. Save this in a file called “flop_inst_clock_node.txt” in the “rb” directory in your home directory. Insert a line with your name at the top of this file.

```
Applications Places System                                   Mon Nov 9, 07:35
adlv30@VLSI18:/cad_area/vlsi/design/rb
File Edit View Search Terminal Help
adlv30@VLSI18 rb]$ grep DFF rb_flat_single_line_adlv30.v | perl -ne '@x = split;print "$x[0] ";print "$x[2]\n" ' > flop_
clock_node_adlv30.txt | more
adlv30@VLSI18 rb]$ gedit flop_inst_clock_node_adlv30.txt
adlv30@VLSI18 rb]$
```

```
flop_inst_clock_node_adlv30.txt (/cad_area/vlsi/design/rb) - gedit
File Edit View Search Tools Documents Help
Open  Save  Undo
flop_inst_clock_node_adlv30.txt
Name - Manjunath Veeraiah Kalmath

DFF_X1 (.CK(mc_rb_ef1_sclk_i),
DFFR_X1 (
DFFR_X1 (.CK(gctl_sclk),
DFFR_X1 (
DFFR_X1 (
DFFR_X1 (.CK(gctl_sclk),
DFFR_X2 (.CK(gctl_sclk),
DFFR_X1 (.CK(gctl_tclk),
DFFR_X1 (.CK(gctl_tclk),
DFFR_X1 (.CK(gctl_core_cclk),
DFFR_X1 (.CK(gctl_core_cclk),
DFFS_X1 (.CK(gctl_core_cclk),
DFFR_X1 (.CK(gctl_core_cclk),
DFF_X1 (.CK(mc_rb_ef1_sclk_i),
DFFS_X1 (.CK(mc_rb_ef1_sclk_i),
DFF_X1 (.CK(mc_rb_ef1_sclk_i),
DFF_X1 (.CK(mc_rb_ef1_sclk_i),
DFFR_X1 (.CK(gctl_core_cclk),
DFFS_X1 (.CK(gctl_core_cclk),
DFFS_X1 (.CK(gctl_core_cclk),
```

c. Using sort and uniq commands (with appropriate options), list each node (net or port) connected to flip-flop clock pins with a count of the number of clock pins it is connected to.

```
adlv30@VLSI18:/cad_area/vlsi/design/rb
le  Edit  View  Search  Terminal  Help
ilv30@VLSI18 rb]$ grep DFF rb_flat_single_line.v | perl -ne '@x=split; print "$x[0]"; print "$x[2]\n"' | sort | uniq
-l

ilv30@VLSI18 rb]$ grep DFF rb_flat_single_line.v | perl -ne '@x=split; print "$x[0]"; print "$x[2]\n"' | sort | uniq
   20 DFFRS_X1(.CK(gctl_sclk),
 1124 DFFR_X1(
 8104 DFFR_X1(.CK(gctl_core_cclk),
  325 DFFR_X1(.CK(gctl_sclk),
  443 DFFR_X1(.CK(gctl_tclk),
    6 DFFR_X1(.CK(mc_rb_ef1_sclk_i),
   19 DFFR_X2(
 1293 DFFR_X2(.CK(gctl_core_cclk),
   41 DFFR_X2(.CK(gctl_sclk),
    2 DFFR_X2(.CK(mc_rb_ef1_sclk_i),
 1114 DFFS_X1(.CK(gctl_core_cclk),
   13 DFFS_X1(.CK(gctl_sclk),
  606 DFFS_X1(.CK(mc_rb_ef1_sclk_i),
  381 DFFS_X2(.CK(gctl_core_cclk),
  103 DFF_X1(
   72 DFF_X1(.CK(gctl_sclk),
  101 DFF_X1(.CK(gctl_tclk),
  704 DFF_X1(.CK(mc_rb_ef1_sclk_i),
  130 DFF_X2(
   29 DFF_X2(.CK(gctl_sclk),
ilv30@VLSI18 rb]$
```

d. List nodes connected to flip-flop reset pins (each node must appear only once in the list) with a count of the number of flip-flop reset pins connected to the node.

1 gctl_sclk_srst_n

13 mc_rb_inporeset_i

3. Write two Verilog modules that instantiate AND and NOR gates to implement the AOI22 function. Call the two modules “SPAOI_X1 and SPAOI_X2. The port names for the modules should be the same as the corresponding pin names in AOI gate. Use X2 drive strength for the AND gates in both modules. Use X1 drive strength for the OR gate in SPAOI_X1, and use X4 drive strength for the OR gate in SPAOI_X2.

a. Copy the netlist “rb_flat_single_line.v “ in your “rb” directory, naming the new file “rb_flat.v”. Append the two modules SPAOI_X1 and SPAOI_X2 to the bottom of “rb_flat_mode.v”