Verilog Gate-Level Netlists and SDC

1) Is the netlist “vorca.v” hierarchical or flat? If hierarchical, how many hierarchical modules are there in the netlist, and what are they called?

The netlist "vorca.v" is hierarchial. Modules are vorca, survik and ICGX1.
command: get_design *

2) How many flip-flops are in the netlist? How many of these flip-flops are connected to the clock port “clk” and how many are not?

i)Flipflop count : 160
ii)flop directly connected to clk port: 8
iii)flop not directly connected to clk port: 152

i)Command:
```
sizeof_collection [all_registers -flop]
```

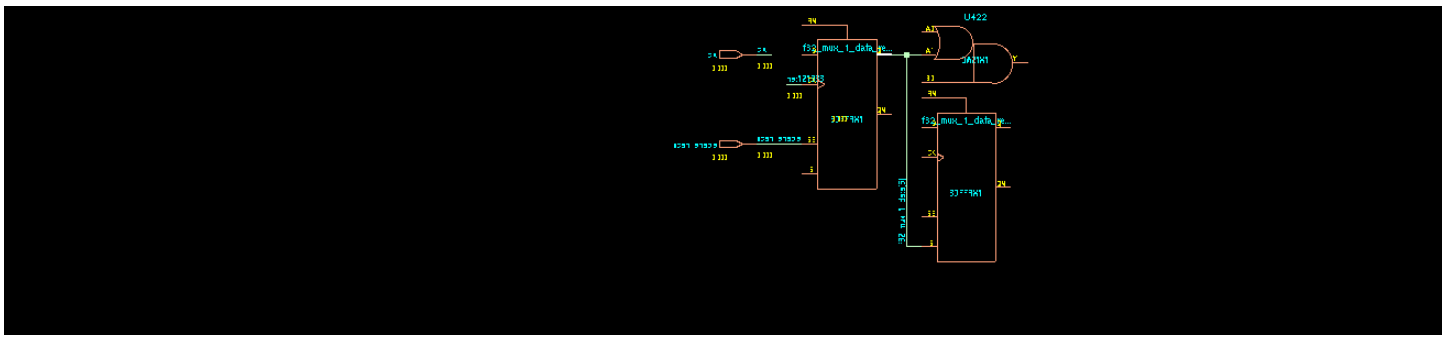
ii)Command
```
set gn [all_registers -latches -clock_pins ]
set full [all_fanout -from clk ]
set reg_gn [all_fanout -from clk -endpoints_only ]
set icg_pin [remove_from_collection $full $reg_gn ]
set level1 [all_fanout -from clk -pin_levels 1]
set inter [remove_from_collection $level1 $icg_pin ]
set final_opt_1 [remove_from_collection $inter $gn]
foreach_in_collection b [remove_from_collection $inter $gn] { puts [get_object_name $b] }
```

iii)Command
```
set all_flop [all_registers -flop -clock_pins]
set final_opt_2 [remove_from_collection $all_flop $final_opt_1]
```

3)Draw a schematic of the fanout cone of the flip-flop “f32_mux_1_data_reg_5_”. How many endpoints are there in this fanout cone? What types of endpoints are they?

```
foreach_in_collection j [all_fanout -from [get_pins f32_mux_1_data_reg_5_/Q] -endpoints_only]
{ puts [get_object_name $j] }
```

output: f32_mux_1_data_reg_6_/SI(INPUT PIN OF REG 6), o_rd_data[5] (OUTPUT PORT)



4)Take one of the endpoints of the fanout cone above and trace its entire fan-in cone.How many startpoints are in this fan-in cone? List all the startpoints, and indicate what type of startpoint they are. Are any startpoints input ports?

foreach_in_collection j [all_fanin -to o_rd_data[5]] { puts [get_object_name $j] }

endpoint : f32_mux_1_data_reg_6_/SI
FANIN:
f32_mux_1_data_reg_6_/SI
f32_mux_1_data_reg_5_/Q
f32_mux_1_data_reg_5_/CK

Endpoint: o_rd_data[5]
Fan_in output: n_write_reg/CK, f_ack_reg/CK, f_state_reg_0_/CK, f_state_reg_1_/CK, f_state_reg_2_/CK, f32_mux_1_data_reg_5_/CK, f32_mux_0_data_reg_5_/CK

The startpoints are clk pin of flops
NO startpoint is input port

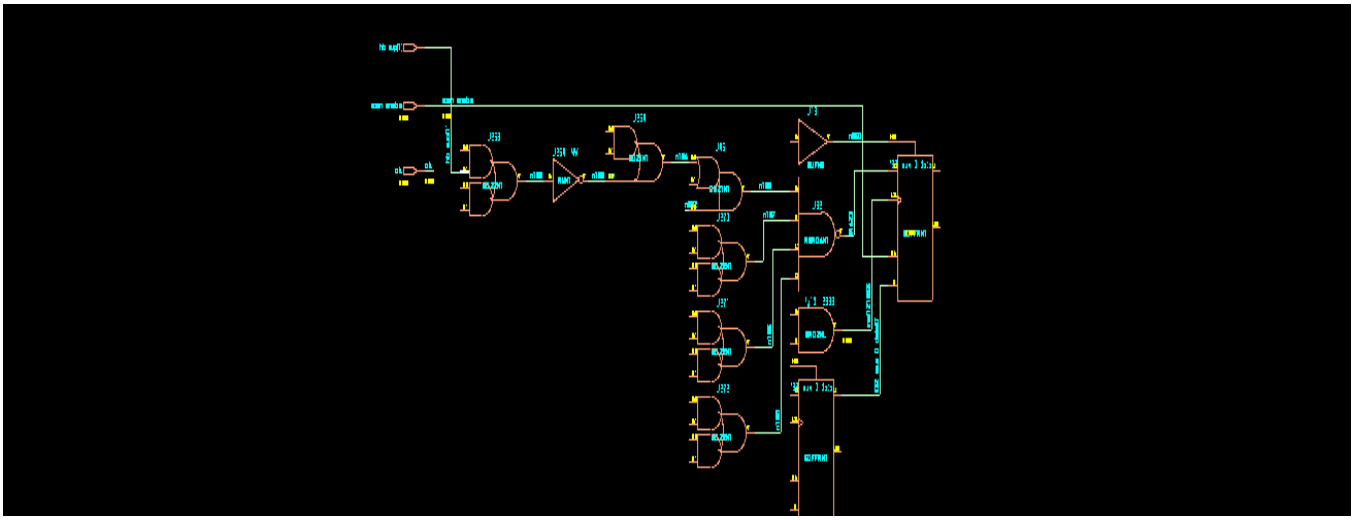
5) There is a three-bit input bus called `i_hb_sup". Trace the fanout cone of bit [1] of this bus and list all endpoints. Submit a schematic of the fanout cone.

foreach_in_collection j [all_fanout -from i_hb_sup -endpoints_only] { puts [get_object_name $j] }
output: f32_mux_0_data_reg_2_/D, f32_mux_0_data_reg_1_/D, f32_mux_0_data_reg_0_/D

foreach_in_collection j [all_fanout -from i_hb_sup[1] -endpoints_only] { puts [get_object_name $j] }
output: f32_mux_0_data_reg_1_/D

foreach_in_collection j [all_fanout -from i_hb_sup[1]] { puts [get_object_name $j] }
i_hb_sup[1]
U263/A1
U263/Y
U264_INV/A
U264_INV/Y
U264/B0

U264/Y
U46/A0
U46/Y
U92/A
U92/Y
f32_mux_0_data_reg_1_/D



6) The clock pins of many flip-flops are not connected to the “clk” port. Where does the clock to these flip-flops come from?

-> The clock to these flipflops comes from ICG Cells.

7) The RAK library does not have an integrated clock gating cell. The vorca netlist, however contains several instances of a clock gater, “ICG”. Create a netlist for ICG with instantiations of actual standard cells and append it to the vorca netlist.

```
module ICGX1(CLK, EN, ENCLK, test_se);
  input CLK, EN, test_se;
  output ENCLK;
  wire CLK, EN, test_se;
  wire ENCLK;
  wire UNCONNECTED, lat, n_0;
  AND2XL g15__2398(.A (CLK), .B (lat), .Y (ENCLK));
  TLATNXL lat_reg(.GN (CLK), .D (n_0), .Q (lat), .QN (UNCONNECTED));
  OR2X1 g17__5107(.A (test_se), .B (EN), .Y (n_0));
endmodule
```

8) Write an SDC for vorca with clock definitions and with input and output delays set at half the clock period. Parameterize the clock period in the SDC, and set its value such that the clock frequency is 1 GHz.

```
create_clock -name sclk -period 1 [get_ports clk]
set_clock_latency -source -max 0.5 [get_clocks clk]
set_input_delay -max 0.5 [get_ports * -filter "direction==in" ] -clock [get_clocks "sclk"]
set_output_delay 0.5 [get_ports * -filter "direction==in" ] -clock [get_clocks "sclk"]
set_clock_transition -rise 0.1 [get_clocks sclk]
set_clock_transition -fall 0.1 [get_clocks sclk]
set_clock_uncertainty -setup 2 [get_clocks sclk]
set_clock_uncertainty -hold 2 [get_clocks sclk]
```