***Question 1.1*** *– Which lines of listing 1.1 and listing 1.2 determine changing d\_in and d\_out at moments of time depicted on figure 1.1?*

***Answer****.*

**Listing 1.1** – Simple D-type flip-flop without reset:

`timescale 1 ns / 1 ps

module dff(clk, d\_in, d\_out);

input clk, d\_in;

output reg d\_out;

always @(posedge clk) begin

d\_out <= d\_in;

end

endmodule

**Listing 1.2** - Testbench for simple D-type flip-flop from code 1.1:

`timescale 1ns / 1ps

module dff\_tb;

parameter period = 4;

parameter seed = 1;

reg clk, d\_in;

wire d\_out;

dff dff\_inst1(.clk(clk), .d\_in(d\_in), .d\_out(d\_out));

initial begin

clk = 0;

forever #(period/2) clk = ~clk;

end

initial begin

d\_in = 0;

@(negedge clk) d\_in = 1;

repeat (20) begin

@(negedge clk) d\_in = $random();

end

$finish;

end

endmodule

***Question 1.2*** *– Explain why flip-flop from listing 1.3 will demonstrate different behavior during simulation and synthesis in situation described in caption of figure 1.2.*

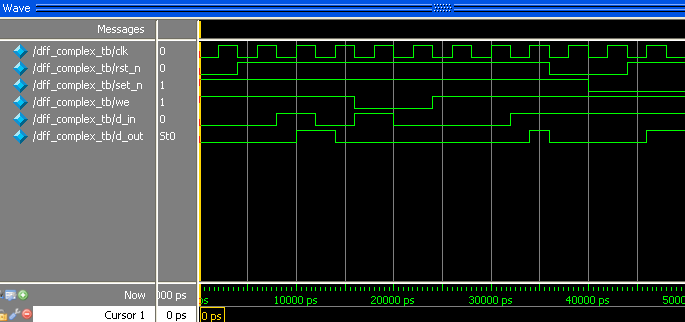


Figure 1.2 – Incorrect simulation of flip-flop with asynchronous reset and set. Flip-flop doesn’t set after transition *rst\_n* to high with low *set\_n*. Setting take place only after positive clock edge.

***Answer.*** *Flip-flop demonstrate different behavior because it operation depends on asynchronous inputs such as set\_n and rst\_n and synchronous input we which can change own states in any moments of time*

***Question 1.3*** *– Explain the function of verilog operators force and release.*

***Answer.*** *A force statement to a variable shall override a procedural assignment or procedural continuous assignment that takes place on the variable until a release procedural statement is executed on the variable. After the release procedural statement is executed, the variable shall not immediately change value (as would a net that is assigned with a procedural continuous assignment). The value specified in the force statement shall be maintained in the variable until the next procedural assignment takes place, except in the case where a procedural continuous assignment is active on the variable.*

***Question 1.4 –*** *Why comparator from listing 1.14 working not correctly with signed numbers?*

***Answer****. Because signed numbers use high bit as signed bit. Listing 1.14 in turn design as unsigned comparator and interprets high bit as bit of positive number as a result any negative numbers are interpreted as positive numbers with set high bit.*

**Listing 1.14** – Comparator of unsigned numbers

`timescale 1ns / 1ps

module comparator(a, b, equal, greater, lower);

parameter width = 8;

input [width-1:0] a, b;

output reg equal, greater, lower;

wire [width:0] tmp;

assign tmp = a - b;

always @(tmp) begin

equal = 0;

greater = 0;

lower = 0;

if(!tmp) begin

equal = 1;

end else if(tmp[width]) begin //checking borrow bit

lower = 1;

end else begin

greater = 1;

end

end

endmodule

***Question 1.5 –*** *Describe disadvantages of pipelining;*

***Answer.*** *Module**executes only one instruction at a sample. Pipelining involves adding hardware to the chip.*

***Question 1.6 –*** *What happens if we increase the number of pipeline stages?*

***Answer.*** *In this case, we obtain grater area of chip but device will operates with grater sample frequency.*