Fork & Synthesis

1) a) #5 $display("ENTRY - 1"); ⇒ 7 ns
#2 $display("ENTRY - 2"); ⇒ 2 ns
#5 $display("ENTRY - 3"); ⇒ 7 ns.
endtask

ENTRY - 2 - 2ns
ENTRY - 1 - 7ns
ENTRY - 3 - 7ns

b) fork
#5 $display(" ENTRY 1"); - 5 ns
#2 $display(" " 2"); - 2 ns
#5 $display(" " 3"); - 5 ns
join
endtask

ENTRY 2 - 2ns
ENTRY 1 - 5ns
ENTRY 3 - 5ns.

3) intial begin
assn a=b+c; = assign can't be there inside initial.
end
Only inside always block.

4) initial begin
initial begin = nested initial can't be used.

5) initial begin
always @(b) begin = always can't be used inside initial.
a = b
end
end

6)

6) initial begin

task add() ; task is declared outside initial block.

endtask

end

7) initial begin

end and g1(a,b,c); ← can't be used initial block.

must be used outside procedural blocks

8) a)

(b) Various procedural blocks = initial - run once

= always - repeat

11) b = will end at 0 ns (0 : ENTRY-1)

c = at 60 ns it ends (50 : ENTRY-2)

d = will end at 5 ns.

12) a) flipflops = 1 for co

1 for s.

b) latches = 1 for co

1 for s.

c) 4 fliflops because : a = 4 bit vector.

Consequential

13) a) always @(sig1 or sig2) begin
end → Latches (O/p changes at sig1/2 changes)

b) always @ (posedge sig1 or posedge sig2)
begin
→ D-FF (O/p changes at posedges.)
end

c)

14) 5 : ENTRY :: 1 @ 5 ns, 15 ns, 25 ns

always block does not start until prev. block ends.

∴ always @(fre edge clk) begin
#15;
$ display (" %t : ENTRY :: 2", $time);
end

∴ 9t occurs at 5+15 = 20 ns.
25+15 = 40 ns
45+15 = 60 ns ....

15)

16) Initial = to initialize the stimulus;
Always = so that as stimulus reg. changes, o/p changes.

17) always begin
clk = 0; #5;
~~clk = 0;~~
~~clk = ~clk;~~ clk = 1; #5;
end

18) Reg = DFF only if its changing at positive edge of flipflop.

→ 7 if changes at both triggering, if is latch.

(a) clk is not initialised.
Simulation must end at some point using $finish(); statement.

20) always @ (**) begin
{c, sf} = a + b;
end.

25) b, c, d. takes value 0, 1, x at end of 10ns respectively.

0ns = b=x c=x d=x
5ns = b=x c=1 d=x
10ns = b=0 c=1 d=x
15ns = b=0 c=1 d=1

26) 0ns = b=x c=x d=x
5ns = b=0 c=1 d=1

27) 0ns = b=x c=x d=x
0ns = " c=1 "
0ns = b=0 " "
" = b=0 c=1 d=1 c finally.

28) 0ns = b=c=d=x
5ns = c=1
10ns = b=0 c=1
15ns = b=0 c=1 d=1

(begin end make the code sequential