Procedural Drill Assignment

3) a) forever #10 clk = ~clk;

→ Only 1 statement, begin and not use

b) To run multiple statements, begin end is required.

Inside which the statements run sequentially one after another.

1xz
= 1x1, 1x0, 1xx, 1z
1?x
= 11x, 10x, 1xx, 1z

4) Casez (inter)

3'b 1?z : y = inter[2];
3'b 01z : y = inter[1];
3'b 001 : y = inter[0];

endcase

5) Practical use of cases :-

In priority encoder
for = interrupt handling
= decoding opcode in CPU
= In state machines.

6) Verilog abstraction levels :- (for testbench)

1) Behavioral (similar to C-programming)
2) Structural / Gate level (include modules in current modules)
3) Dataflow (directly assigned by assign)

for design

8) Race condition occurs whenever 2 events execute at the same same scheduling region in Verilog, the o/p of the code varies based on compiler (inconsistant result).

→ This is called racing.

→ That is, the o/p cannot be determined and varies b/w various possible outputs.

Ex - initial begin
a <= 23;
and b <= 32;
and initial begin
a <= b;
b <= a;
end

at 0ns,
a = 23, b = 32
Also at
0ns, a <= 32; b <= 23.

Here, o/p = 23 & 32 when $display.
= 32 & 23 when $strobe/monitor

9)

Blocking assgn	Non-blocking assgn
→ Execute sequentially one after another. (Execute & assign).	→ Execute concurrently. → Assign concurrently in the NBA region.
→ Happens in active region.	→ Used in sequential circuits where o/p depends on clock.
→ Used in combinational ckt where o/p needs to change immediately.	

10)

```
always ——————
    Q1 = d;
    Q2 = Q1;
    Q3 = Q2;
end
```

= 1 dff

11)

```
always ——————
    Q1 <= d;
    Q2 <= Q1;
end Q3 <= Q2;
```

= 3 dff

12) In Non-blocking as the execute & assignment happen concurrently, at posedge clock, & having the previous value of signals, changing the order does not change the output.

13) If sensitivity list has pos/neg edge and using Non-blocking statements infer a separate flip flops and assignments occurs at the specific instance, that in the actual hard

Without such sensitivity list, the o/p needs to come immediately as i/p changes, hence use blocking assignment instead.

15) 3 Diffs -

→ There is no dependency

∴ Order of placing statement won't matter even if it is blocking statement.

```
@ (posedge clk)
Q1 = d;
Q2 = a;
Q3 = b;
```

16) 10 synthesizable coding construct :-

operators - +, -, &, |, ^, ~

data types - wand, wor, wire, reg[7:0], arrays

- if, else, case, etc.
- and, or, nor, xnor.

basic - module, endmodule, function, event, @(posedge clk) - negedge/posedge.

- always
- begin, end
- blocking, non-blocking, 2'b11, etc.

parameter, compiler directive

10 non-synthesizable (fork, join, force, release)

→ System task, real constants, time, event, inbuilt gate delay, UDP, table, ===, !==, →, →>, delay, wait (#), initial

17) - 5 dff (5 bit variable)

{c, s} = a + b + cin

(o/p delayed by 1 clk cycle)

18)

1 dff of y.

z = x & y. = reg.

20) a) CDC is technique of passing a signal from one clk domain to another in such a way that it doesn't result in metastable output.

b) SoC have various domains connected to it, each operating at a different clock according to their needs. These modules are made by different organisation as well.

(c) We use a synchroniser (which is like a shift register (2FF)) that gives necessary delay to signal to settle down and cross to next clock domain without glitches. Thus preventing metastability.

21) Timescale has 2 components

Timestep — Timeprecision.

b) `timescale 10ns/1ns

c) `timescale 1ns/1ns.

i) During clock generations,

if we want a tp of 11.

∴ tp/2 = 5.5 is

round upto 6.

∴ O/p tp = 12.

d) i) `timescale 1ps/1ps

└ cause too much computational load.

ii) Slow code ~~execution~~ & simulation but highly precise simulation.

22) Interdelay statement — the delay occurs after the end or in the beginning of statement execution & assignment.

Intradelay — delay occurs in the middle of statement execution & assignment.

b) Delays are non-synthesizable, hence not applicable for RTL design.

c) #10 a = 10; - inter delay.

d) a = #10 10; - intra delay.

2) a) System task is a predefined task in verilog that executes a particular operation.

Ex - $value$plusargs ( ) — takes user arg.

$finish( ); — end simulation.

$display( ); — print o/p in transcript

c) Various categories of system task :-

(1) takes user arg = $value$plusargs

(2) $finish / $stop = ends / stops simulation

(3) $display / $monitor / $write / $strobe = write msg on trans

(4) $feof = find end of file
$fclose = close file.

(5) backdoor = $readmemh / $readmemb
$writememh / $writememb

h) Display window default = transcript.

k) i) $stop = stops the program (used for debugging). (break point)

$finish = end the program.

ii) Breakpoints are useful in understanding changes to program o/p in each line of code as simulation progresses.

(i) $write

=> prints in same line as last line.

$write ("1");
$write ("0");

o/p = 10

$display

=> prints in new line after being called.

$display ("1");
$display ("0");

o/p = 1
0

j) $strobe

=> executes at the end of timestep in scheduling region (in monitor region)

$write

=> executes at active region.

Ex. a = 10;
a <= 5;
$strobe ("%d", a);
$write ("%d", a);

o/p = 5 - strobe
10 - display.

k) $time

-> $t return unsigned integer type output.

Ex. 1ns, 2ns, etc.

-> less precise

$realtime

-> $t returns float type timestamp of the current time

Ex - 1.3 ns, 2.16 ns

-> more precise.

24 - b Seed makes $random produce different o/ps for various all different seed i/p. Hence various kinds of & stimulus can be generated.

25) a) Compiler directive is a macro that is used to instruct the compiler on how the simulation shall run. keyword

b) Various compiler directives are :-

- `timescale — how simulation must run wrt time.
- `define - holds constant / expression. (increase reusability of code)
- `ifdef / `endif / `elsif — can be used to implement feature selectively in design
- `include : import other files to current file

c) Macro BUS / `BUS increases reusability & of datatype declaration. Name coding convention.

d) Two ways of macro definition -

(1) `define BUS reg[7:0]
module -

endmodule
or

(2) vlog +b +define+BUS=reg[7:0]

→ If declared simultaneously, the one used in run.do file get executed & other gets overridden.

e) Macro

→ Macro like `define can be used to replace of parameter.

→ They support constant, as well as expression.

Parameter

→ Parameter can be declare used to constants in code.

→ They make use of these constant reusable.

Parameters can be easily overridden in testbench. Hence for verification, they are preferrable.