Solver: Zhao Peizhu

1. Multiple choice

(1) Ans: **A** – By repeatedly multiplying 16(0x10000), we can obtain 4 bits of answer per multiplication.
1st 4 bits – $0.3456 \times 16 = \mathbf{5}.5296, 5 - 0101_2$
2nd 4 bits – $0.5296 \times 16 = \mathbf{8}.4736, 8 - 1000_2$
3rd 4 bits – $0.4736 \times 16 = \mathbf{7}.5776, 7 - 0111_2$
4th 4 bits – $0.5776 \times 16 = \mathbf{9}.2416, 9 - 1001_2$

(2) Ans: **E** – The answer can be seen from the computation directly.

	1	0 1 0 1 0 0 0
+	1	1 1 0 0 1 1 1
1	1	0 0 0 1 1 1 1
carry	No OF	

(3) Ans: **B** – The answer can be seen from the computation directly.

	0	0 1 1 1 0 0 1
+	0	1 0 0 1 0 1 1
0	1	0 0 0 0 1 0 0
No carry	OF	

(4) Ans: **D** – The answer can be seen from the computation directly.

A.	0	0 1 0 0 0 0 0
-	1	1 0 1 0 1 0 1
0	0	1 0 0 1 0 1 1
No carry	No OF	

B.	0	1 1 0 1 1 0 0
-	1	1 1 1 0 0 0 1
0	0	1 1 1 1 0 1 1
No carry	No OF	

C.	1	1 1 1 0 0 0 0
-	0	0 1 0 1 0 0 1
0	1	1 0 0 0 1 1 1
No carry	No OF	

D.	1	0 0 0 0 0 1 1
-	0	1 1 0 1 0 1 0
0	0	0 0 1 1 0 0 1
No carry	OF	

(5) Ans: **C** – For W to drive X, the parameters of W and X should fulfill:
a. $V_{IH,min,X} \leq V_{OH,min,W}; V_{IL,max,X} \geq V_{OL,max,W};$
b. $V_{IH,max,X} \leq V_{OH,max,W}; V_{IL,min,X} \leq V_{OL,min,W};$
From these 2 relations we can directly eliminate other options.

(6) Ans: **D** – $Margin = \min\,[(V_{IL} - V_{OL}), (V_{OH} - V_{IH})]$. By computation for each option and calculate the maximum of these options, we choose D.
A. $Margin = \min[(1.2 - 0.9), (3.8 - 3.5)] = 0.3$
B. $Margin = \min[(1.3 - 1.0), (3.2 - 2.8)] = 0.3$
C. $Margin = \min[(2.0 - 1.4), (3.6 - 3.2)] = 0.4$
D. $Margin = \min[(1.7 - 1.2), (4.0 - 3.5)] = \mathbf{0.5}$

E. $Margin = \min[(1.5 - 1.1), (3.5 - 3.0)] = 0.4$

(7) Ans: **B**

$$[(a' + b)c'd + (a'c)' + b'd']' = [(a' + b)c'd]' \times (a'c) \times (b'd')'$$
$$= [(a' + b)' + c + d'] \times (a'c) \times (b + d)$$
$$= (ab' + c + d')(a'c)(b + d)$$
$$= (aa'b'c + a'c + a'cd')(b + d)$$
$$= [a'c(1 + d')](b + d)$$
$$= a'c(b + d)$$
$$= a'bc + a'cd$$
$$= m(3,6,7)$$

a'bc – 011X → 0110(6) and 0111(7)
a'cd – 0X11 → 0111(7) and 0011(3)

(8) Ans: **A** – From the provided information, we can deduce the following:
G* is asserted (G = 0), when
1. A* is asserted (A = 0), C* is asserted (C = 0), B* is negated (B = 1)
2. A* is asserted (A = 0), C* is asserted (C = 0), D is negated (D = 0)
From [1], 010X → 0100(4) and 0101(5)
From [2], 0X00 → 0000(0) and 0100(4)
∴ M (0, 4, 5) and answer is **A**.

(9) Ans: **D** – Open drain output cannot connect to normal outputs; e.g. The output of a not gate, since such connections can create short circuit. (i) is wrong. For open drain, being Hi-z state when enable is negated is the characteristic of open-drain output. (ii) is correct. Open drain must be used with pull-up resistors to protect the circuit from short circuits. (iii) is correct.

(10) Ans: **E** – Schmitt trigger inverters have the following characteristics:
a. It has 2 boundaries: voltage V_{T+} and V_{T-}, and $V_{T+} > V_{T-}$
b. When the voltage falls below V_{T-}, the output is set to high
c. When the voltage rises above V_{T+}, the output is set to low
d. Hence it never has indeterminate state
Hence, all statements are false.

(11) Ans: **D** – The operator "&" means bitwise "and" logical operation.

(12) Ans: **D** – In Verilog, the real part and the sign part is separately declared, i.e. the declaration -4d'3 is legal.

(13) Ans: **E** – A and D are correct while B and C are wrong.

(14) Ans: **B** – This is the characteristic of Verilog synthesizer.

(15) Ans: **C** – The most standard form to extend bits (or, repeat bits) is
{number{dummy_signal}}
A is wrong since the most outside layer of brackets should be a pair of braces. 3(x[7]) in B means multiplication. C is the most standard form so C is correct.

(16) Strictly, there is no correct answer. Both A and B are correct as you see in your lab classes. (A design can have only 1 module, and a module can have only one always block.) C is wrong because the content in all always blocks are continually monitored and refreshed. D is correct, because combinational always blocks refer to pieces of combinational circuits. Hence all A, B, D are correct.
Additionally, I had asked prof about the option E and the prof explained that E means all A, B, C, D are wrong.

(17) Ans: **B** – It is very important to distinguish two operators: = and ==. = is a non-blocking assignment while == means comparation (the output is and only is 1 when LHS = RHS). Hence a==2'b11 is false, the expression y=0 gives an output 0.

(18) Ans: **E** – Remember that a module cannot be instantiated within any always block.

(19) Ans: **A** – B[0] is 1'b1, C[1] is 1'b1, and the pair of braces means signal extension.

(20) Ans: **D** – This is the characteristic of JK flip flops.

2.

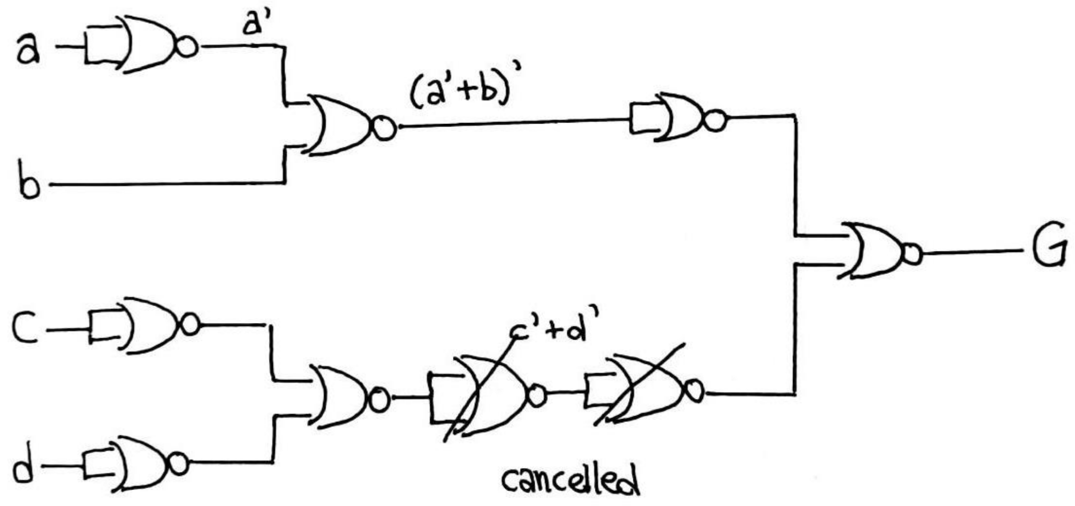
(a)

(i) $+74 = 64(1) + 8(1) + 2(1)$
$= 2^6(1) + 2^5(0) + 2^4(0) + 2^3(1) + 2^2(0) + 2^1(1) + 2^0(0)$
$= 01001010_2$
To get -74 2's complement, invert and add 1 to the binary:
01001010 → 10110101 → **1011 0110**

(ii) $+38 = 32(1) + 4(1) + 2(1)$
$= 2^5(1) + 2^4(0) + 2^3(0) + 2^2(1) + 2^1(1) + 2^0(0)$
$= \mathbf{0010\ 0110_2}$

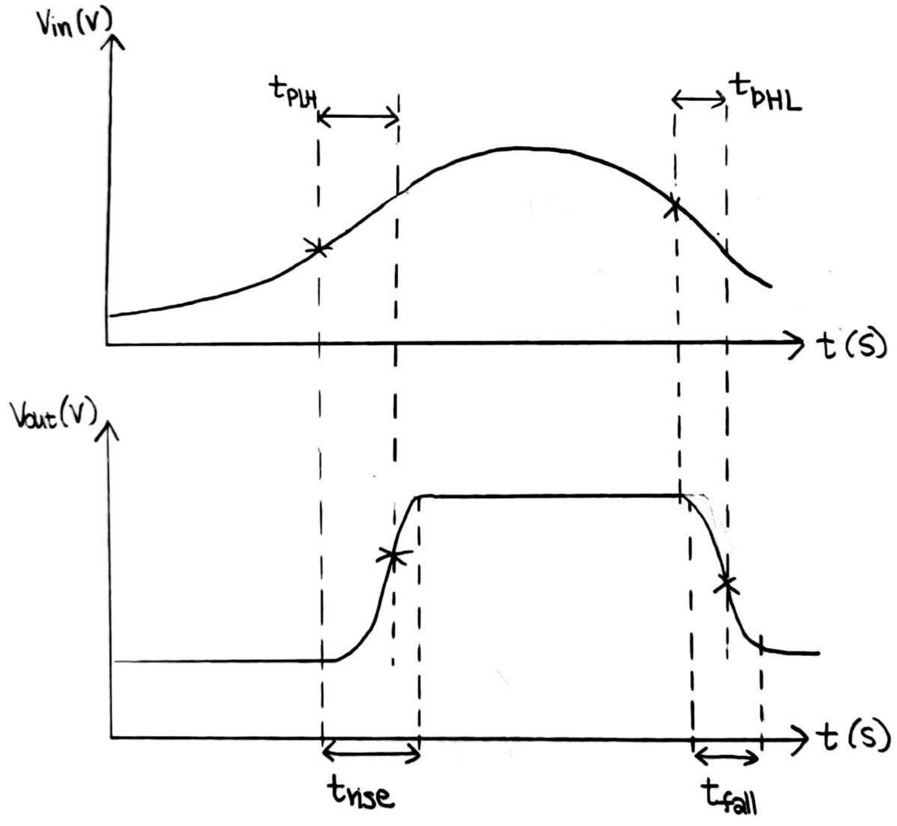
(b)
$$\begin{aligned}F(w,x,y,z) &= [z'(w+x+y')' + xyz(wz)' + wx'y'z']' \\ &= [z'(w+x+y')']'[xyz(wz)']'[wx'y'z']' \\ &= [z'(w+x+y')']'[xyz(w'+z')]'[wx'y'z']' \\ &= [z'(w+x+y')']'[w'xyz + xyzz']'[wx'y'z']' \\ &= [z'(w+x+y')']'[w'xyz]'[wx'y'z']' \\ &= (\boldsymbol{z+w+x+y'})(\boldsymbol{w+x'+y'+z'})(\boldsymbol{w'+x+y+z})\end{aligned}$$

(c) $(cd)' \equiv c' + d$
$Total\ prop.delay = the\ delay\ on\ the\ longest\ path\ (from\ a\ to\ G) = 4T$



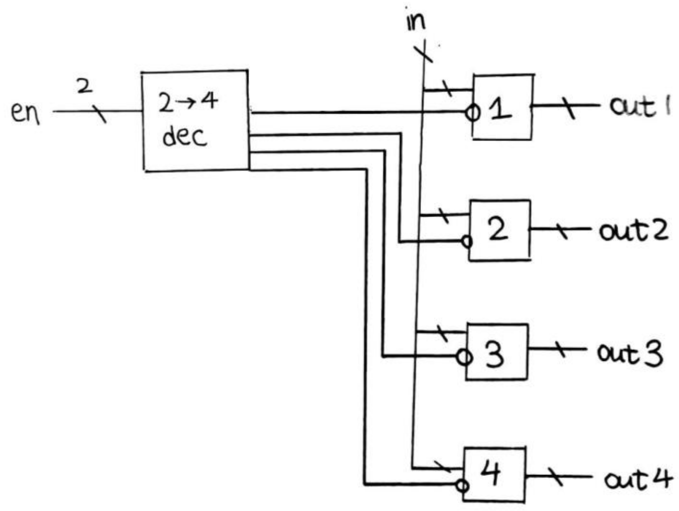
(d) t_{PLH}: the propagation delay for V_{out} to change from L to H. t_{PHL} is similar

For example, for a buffer,



(e) 1. The components can be disabled.
2. Only one can be enabled at a time, if there exist enabled components.

∴ We should use Hi-z.
To save input we use a decoder. (Notice that enable* is an active-low signal)



(f) We can figure out pieces of K-map from all cases given directly, listed as follows:

AB \ CD*	10
11	1
10	1

AB \ CD*	10
01	1
11	1

AB \ CD*	00	10
11	1	1

AB \ CD*	00	10
11	1	1

AB \ CD*	01
00	0
01	0
11	0
10	0

AB \ CD*	00	01	11	10
00	0	0	0	0

Hence the entire K-map is as follows:

AB \ CD*	00	01	11	10
00	0	0	0	0
01	x	0	x	1
11	1	0	1	1
10	x	0	x	1

With four 2*2 loops we can obtain the minimum SOP:
Y = BC + BD*’ + AC + AD*’.
Do not forget to add a star to denote that D* is an active low.

3.

(a)

(i) Notice that, from the characteristics of the decoder, we can figure out the SOP form of X and Y.
X = m (1, 2, 4, 7)
Y = m (3, 5, 6, 7)

From a K-map or from a truth table, by observation (I cannot figure out other better method. At least observation is useful for dealing with only 8 cases.), we can figure out that X is an Xor gate while Y is a majority gate. (Please see the tutorial for the majority gate.)

(ii) Assign {X, Y} = {a? b^c: ~(b^c), a? b|c: b&c}
Other parts are omitted. Here I use ‘?:’ statements to simplify the code.

(b)

(i)
```
always @ * begin
   case (st)
         2’b00: if (I == 1) nst = 2’b01; else nst = 2’b00;
         2’b01: if (I == 1) nst = 2’b10; else nst = 2’b00;
         2’b11: if (I == 1) nst = 2’b10; else nst = 2’b00;
   endcase  end
```
This is a very straight forward answer. Several utilizations on the code can be taken.

(ii)

St[1][0]	I[0]	Nst[1][0]	Output[0]
00	0	00	0
00	1	01	0
01	0	00	1
01	1	10	1
10	0	00	0
10	1	10	0

(iii) From the diagram in (ii), we can have K-maps for output variables:

	S[1]S[0]			
	00	01	11	10
0	0	1	x	0
1	0	1	x	0

		S[1]S[0]			
		00	01	11	10
N[1]N[0]	0	00	00	x	00
	1	01	10	x	10

Hence, we have:
Out = S0
N1 = IS0 + IS1

N0 = IS1'S0

(iv) From (iii):

```
assign P = S0;
assign N1 = I&S0|I&S1;
assign N0 = I&~S1&~S0;
```

(v) The approach using case statement is very straightforward and the readability of the code is strong hence it can spread very fast and be used very efficiently. The approach using assign statements save some lines of code but the readability is affected badly. Actually these are the differences between behavioural/objective approach (using case statements) and functional approach (using assign statements).

--End of Answers--