EEE4120F Quiz 3 based on:

Lecture 14 - 17 and Verilog Essentials

DATE: 20/4/2023

ANSWERS!!!

Please fill in name!

*This is a short quiz, but it **is** for marks!*

NB: Please select only one answer option for each question

CIRCLE/COLOR-IN ANSWERS FOR MULTIPLE CHIOCE QUESTIONS

TOTAL NUMBER OF QUESTIONS : FIVE (5) **TIME (mins):** **15**

#	Question - EACH QUESTION WORTH 1 MARK	Sec	W	%	X
Q1	A PLA and a CPLA is not exactly the same thing. It was very briefly indicated what a CPLA or 'midrange' type of programmable logic item is … which one of these most accurately describes what a CPLA is	80	2	10%	
	[1] A CPLA is the same a a FPGA, programmed and structured the same. [2] A CPLA actually a PLA, also just an array of the same gate types but the way it is programmed is a bit different. [3] A CPLA is a more advanced technology than an FPGA. **[4] A CPLA is more sophisticated than a PLA as it is an interconnected set of PLAs in which these interconencts can be configured.** [5] The difference is essetnially just in speed, the CPLA is much faster.				
Q2	I discuss PLBs and PLEs quite a bit. But how are these two things related? Select the best answer below.	80	3	15%	
	[1] A PLE manages (via configuration MUXes) one or more PLBs. [2] A PLE is a more advanced circuit element than a PLB. [3] Both a PLE and a PLB are equivelent in logic, but the PLB is faster. **[4] One of more PLEs, together with configuration MUXes, would be contained within a PLB.** [5] The main thing is that you should always have the same number of PLBs as PLEs for a logic design to be executable on an FPGA.				
	CODE TO VIEW FOR Q3-Q5 : Here's a simple Verilog code module. Take a look over that and answer the questions that follow. Edit the code below directly to respond to Q3				
Q3	*// Simple Verilog code module. Rather lack of comments.*	320	5	25%	

```
// 4-bit down-count module (downcount is a metter name
for this module)
module test (input clk, rst, output [3:0] xout );
 reg [3:0] x; // internal counter register
 // do work only when positive clk or rst
 // await a positive edge on clk or rst
 always @(posedge clk or posedge rst)
 begin  // force register x to 0xF on reset
  if(rst) x <= 4'hf;
 else   // otherwise if reset not high decrement
  x <= x - 4'd1;
 end
 assign xout = x;  // link output x to value of xout
endmodule
```

Q3	Todo for Q3: look over the code on the previous page. Briefly explain in English what functionaliy this module is implementing. (5 marks) The test module is implementing a 4-bit down-counter with reset. The reset is performed if there is a positive edge on either clk or the rst line itself. On reset, the counter x register is forced to 0xF (15 in decimal). On each clk, wne when rst is not high, the x counter register is decremented by 1. There is no limit checking, the x value will wrap round to 0xF if decremented from 0. The internal x register is linked to the xout output bus port.				
Q4	You would probably agree with me in saying that Verilog code is pretty poorly commented. Not to mention lazy naming of a potentially important ports. Go ahead and add improved comments. And by there way, there might be a syntax or typo in that code, see if you spot it and fix it (comment in the line below if you'd like to mention the error or assure that there is no error).	210	5	25%	
	comment re error: See red comments above for added comments. The error is an semicolon (;) missing after 4'hf which has been added in red above.				
Q5	Another thing to do regards that test module (I'm sure chuffed with myself for giving the module a meanigful name albit withotu describing its function). But now you need to inspect some more code. Below is a test bench for the test module. Write down in the space indicated what the first three lines generated from the $monitor operation would display (you don't need to get it perfectly right, a suitably close answer would get full makes).	210	5	25%	

```
// testbench for mystry module#1
module test_testbench();
  reg clk, rst;
  wire [3:0] xout;
  // device (besides the student) under test ...
  test dut(clk, rst, xout);

  initial begin
    rst=1; clk=0;
    $monitor("clk=%b rst=%b xout=%d",clk,rst,xout);
    #10;
    rst=0;
    repeat (10)
        #5 clk = !clk;
  end
endmodule
```

Show what (at least) the first three lines displayed to the log by $monitor will look like (if you want to be fancy you can go beyond 3 lines but it won't necessarily get you any bonus marks):

You can view and run the code on EDAPlayground at:
https://www.edaplayground.com/x/F9Xg

Results is (i.e. using iVerilog):

clk=0 rst=1 xout=15
clk=0 rst=0 xout=15
clk=1 rst=0 xout=14 } these first three lines were expected as an answer
clk=0 rst=0 xout=14
clk=1 rst=0 xout=13
clk=0 rst=0 xout=13
clk=1 rst=0 xout=12
clk=0 rst=0 xout=12
clk=1 rst=0 xout=11
clk=0 rst=0 xout=11
clk=1 rst=0 xout=10
clk=0 rst=0 xout=10
Done

TOTAL :	**900**	**20**	**100%**

Time : time est. in sec W : Weighting of question % : How much question counts X : Office use