University of Tehran
Electrical and Computer Engineering Department
Digital Systems Design With VHDL, Test # 2
ECE
Fall Semester 1391

Computer Account #____________________

First Name :__________________________

Last Name :___________________________

Student Number :_______________________

Signature :___________________________

Grade:

Problem 1.	______/17
Problem 2.	______/17
Problem 3.	______/17
Problem 4.	______/17
Problem 5.	______/30

Total: ______/98 (2 free points / 100)

ONE PAGE PER PROBLEM
THIS IS AN OPEN BOOK EXAM
USE std_logic_1164 FOR ALL DESIGN PROBLEMS
USE OF SIMULATION PROGRAMS IS NOT ALLOWED
YOU HAVE A TOTAL OF 180 MINUTES FOR THIS TEST

1. In the following partial codes, signal *count* which is of type BIT becomes 1 at time 13 and 0 at time 55. Write values of *a* and *b* between time 0 and 40. Put dots in unused rows. Assume 0 initial values. Variables *a* and *b* are INTEGER.

```
-- code 1
PROCESS . . . BEGIN
  WAIT UNTIL count = '1';
  WAIT FOR 12 ns;
  a := a + 1;
END

-- code 2
PROCESS . . . BEGIN
  WAIT ON count;
  WAIT FOR 9 ns;
  b := b + 1;
END
```

Time	**Count**	**a**	**b**
00	0		
13	1		
	1		
	1		
	1		
	1		
	1		
	1		
	1		
	1		
	1		
55	0		
	0		
	0		
	0		
	0		
	0		
	0		

2. Show waveforms applied to the CUT inputs up to 80 NS.

```
ENTITY tester IS END TESTER;
--
ARCHITECTURE data OF tester IS
  SIGNAL reset, clock : BIT := '1';
  SIGNAL xx : BIT_VECTOR ( 3 DOWNTO 0) := "1101";
  SIGNAL z : BIT;
BEGIN
  CUT : CircuitUnderTest PORT MAP( xx, reset, clock, z );

  PROCESS BEGIN
    reset <= '1';
    WAIT FOR 26 NS;
    reset <= '0';
    WAIT FOR 4 NS;
    WAIT;
  END PROCESS;
  --
  clock <= NOT clock AFTER 5 NS WHEN NOW <= 70 NS ELSE '0';
  z <= xx'STABLE(21 NS)'TRANSACTION;
  --
  PROCESS BEGIN
    WAIT UNTIL clock = '1';
    WAIT FOR 3 NS;
    xx <= xx(0) & xx(3 DOWNTO 1);
  END PROCESS;

END ARCHITECTURE;
```

3. In the VHDL code shown below, four processes generate drivers on *w*, *x*, *y*, and *z* signals. Show transactions that expire on the drivers of each signal.

```
PACKAGE placing IS
   TYPE four IS (a, b, c, d);
END placing;

USE WORK.placing.ALL;
--
ENTITY placement IS END ENTITY;
--
ARCHITECTURE sequential OF placement IS
   SIGNAL w, x, y, z : four;
BEGIN
  w <= four'RIGHTOF(w) AFTER 100 NS
       WHEN (w /= four'RIGHT AND NOW <= 101 NS)
       ELSE a AFTER 100 NS;

  x <= four'LEFTOF(x) AFTER 80 NS
       WHEN (x /= four'LEFT AND NOW <= 101 NS)
       ELSE d AFTER 80 NS;

  y <= a AFTER 20 NS, w AFTER 30 NS, x AFTER 40 NS;

  PROCESS (x) BEGIN
     z <= TRANSPORT w AFTER 20 NS;
  END PROCESS;
END sequential;
```

4. Show synthesizable VHDL code for a register unit that performs operations shown below. The unit has a 3-bit mode (*md*) input, an asynchronous reset (*rs*) input, a 1-bit output tri-state control (*oe*) input, and an 8-bit bi-directional *io* bus. The internal register drives the *io* bus when *oe* is 1 and *md* is not 111. Use *std_logic_1164*.

```
md=000: does nothing
md=001: right shift the register
md=010: left shift the register
md=011: up count, (000, 001, 011, 010, 110, repeat)
md=100: down count, (opposite of above)
md=101: complement register contents
md=110: swap right and left 4 bits
md=111: parallel load

Provide asynchronous active high reset.
Positive edge clock
```

5. Write a memory module that is shared and accessed by several other modules. The memory is arbitrated for being accessed by the modules that perform memory block read or write operations. When a module needs to read or write a block from or to the memory, it asks checks if the memory is free and can be accessed. If so, the specified operation that may take several nano-seconds will take place. If the memory is busy with another device, the read or write operation goes on hold until the memory completes its pending operation. Devices will be blocked if they need to access the memory and the memory is busy with another device (read or write module).

   a. Show the block diagram of this environment using four read-write modules
   b. Implement the memory using sc_mutex. The memory is to be declared as an 8-K byte memory.
   c. Write a read-write device (SC_MODULE) that requests memory access at random times, reads or writes a block of memory in a given time, gets busy with other things (mimic this by simple wait statements), and then again tries to access the memory. Make these as parameters that the module is configured by: reading or writing, memory block size, starting address, and the time the device is busy with other things. The time it takes the memory to perform the read or write operation depends on the size of the block, and is handeled by the memory.