Summer 1999 – Test 1 Solutions

1. (10 pts) Write a process that will function as a clock doubler. The sensitivity list should trigger on a signal named *'clk*' and the process should drive a signal called '*clk2x'*. You can assume that the input signal is of type 'std_logic' and that it will only have values '1' and '0'. You can assume that the frequency of *clk* will not change over time and that the duty cycle of *clk* is 50%. You DO NOT know the frequency of '*clk*' ahead of time. The frequency of *clk2x* should be twice that of *clk*, and have a 50% duty cycle.

```
process (clk)

variable half_per: time;
begin
    clk2x <= transport '1',  transport '0' after clk'delayed'last_event/2;
end process;
```

2. (15 pts) a. Draw the waveform generated by the following code:

```
architecture A  of E
begin
signal  a: std_logic = 'Z';
 process
  begin

    a <=  transport 'L'  after 5 ns;
    a <=   transport '1'  after  10 ns;
    a  <=   transport  'H' after  2 ns;
  end process;
end  E;
```

b. Draw the waveform generated by the following code:

```
architecture A  of E
begin
signal  a: std_logic = 'Z';

    a <=  transport 'L'  after 5 ns;
     a <=   transport '1'  after  10 ns;
    a  <=   transport  'H' after  2 ns;

end  E;
```

3. (10 pts) Write a process that triggers on two signals, ‘clk’ and ‘a’. Both are std_logic type and will only be set to the values ‘1’ or ‘0’. Have the process print the setup and hold times of signal 'a' referenced to the **rising** edge of the 'clk' signal.

```
process (clk, a)
 variable  setup, hold: time;
begin
    if (clk'event and clk = '1') then
      -- rising edge
       setup :=  a'last_event;
      assert false
         report "Setup time is: " & To_String(setup) & LF;
    end if;

    if (a'event and clk = '1') then
      hold := clk'last_event;
      assert false
         report "Hold time is: " & To_String(hold) & LF;
    end if;

end process;
```

4. (15 pts) Explain what each of the following attributes return (you MUST give the data type that is returned as well as explain what it is).

   a. 'transaction - *returns a 'boolean signal' that toggles every time an assignment is made to the original signal.*

   b. 'event - *returns a boolean type; true if a change in signal value has occurred.*
   c. 'last_event - *returns a 'time' type; returns the amount of time from NOW until the last change in signal (NOW - time of last event)*

   d. 'range - *returns a range which has two integer endpoints; gives the high and low indicies of a vectored signal.*

   e. 'active - *returns true if an assignment has been made to this signal (if a transaction has occurred). at this delta time point.*

5. (50 pts) Answer the following short answer questions:

a. Given an std_logic_vector, write a VHDL code fragment that will return the number of bits in the vector.

```
variable  k: integer

k :=  signal_name'length;
```

b. Given a std_logic_vector, write a VHDL code fragment that will examine each bit of the vector and replace U, W, ‘-‘ values with an ‘X’.

```
for i in  signal_name'range loop
      if (signal_name(i) = 'U'  or signal_name(i) = 'W' or signal_name(i)= '-') then
         signal_name(i) = 'X';
end loop;
```

c. If the current time is ’10 ns’, what gets entered on the time queue (value and time) for the following assignment:

a <= transport ‘0’;

10 ns + 1delta, '0'

e. If an initial value is not specified for a signal or variable of ANY datatype, what is the default initial value? *leftmost value*

f. What types of statements (sequential or concurrent) can be placed within a GENERATE block?

concurrent

g. What is an advantage of placing a CONSTANT assignment within a package body?

A Constant within a package body breaks dependencies between the package and any entities that use this constant (the constant value is assigned at elaboration time). If you change the constant, you do NOT have to recompile the entities that use this package. If the constant is declared within the header, then changing the constant will force you to recompile the entities that use the package.

h. Why would we want a function to be executed at ELABORATION time instead of during simulation? (Hint: we looked at an example in class where this was necessary).

If you are building some structure via GENERATE blocks and the structure is parameterized based upon some data in an external file. The GENERATE blocks are expanded at elaboration time so you need the external data read in at elaboration time. This is how the PLD models were written (the Jedec file contents parameterized the structure that was built during elaboration).

i. What is meant by *function/procedure overloading*? Give an example of a standard VHDL function or procedure that is overloaded.

When multiple versions of a function/procedure exists; with each version differentiated only by its argument list. The WRITE, READ functions in the textio package are overloaded.

j. Explain what will happen in the following code; explain what happens on the time queue, and what happens within the simulator. Give the final value of 'a'.

```
signal a : std_logic;


process (a)

 begin

    a <= '1';
   if (a = '1') then
        wait;
    end if;
   a <= '0';

  end if;
```

The '1' assignment to 'a' will not take place until the process is exited; so the 'if' test will fail. The last assignment to 'a' (the '0' value) will take precedence, so the final value of 'a' will be a '0'. The process will trigger two times (once when 'a' is initially 'U', once at 0ns + 1 delta when a is '0').

k. What is the purpose of VHDL configurations?
Allows the user to specify what architecture to be used with an entity, also allows specification of generic values.

6. (15 pts) Assume that I am trying to model a single wire network using the ethernet protocol. An ethernet cable can have multiple drivers, and uses a packet collision detection method in which a driver listens to the packet it just sent - if the packet is garbled, then the driver assumes a 'collision' occurred and the driver waits a random amount of time (greater than a packet send time) before trying to send a packet of data again.

If you were using VHDL to model this system, what sort of approach would you use for the data types? (a packet can hold lots of information such as number of bits in the packet, checksum fields, data fields, etc). Explain how you might implement the collision detection feature of this system.

I would use a RECORD type to define the various packet fields within an ethernet packet. The record type could be something like:

```
TYPE  packet IS
  RECORD
    checksum:  integer;
    numbits:    integer;
    sender_address:  integer;
    recv_address : integer;
    collision :   boolean;        -- indicates if a collision has occurred.
 END RECORD;
```

I would make the 'packet' data type a RESOLVED data type since there will be multiple drivers on the ethernet cable; the resolution function will be responsible for detecting packet collisions within a particular time interval. If a collision occurs, then the resolution function will place a packet on the bus indicating a collision; the hosts on the ethernet will be responsible for listening and detecting this collision and performing a randomized exponential backoff algorithm.