Spring 2002 – Test 1 Solutions

SSN: _____ (no names please)

You may only use the VHDL Language Reference Manual

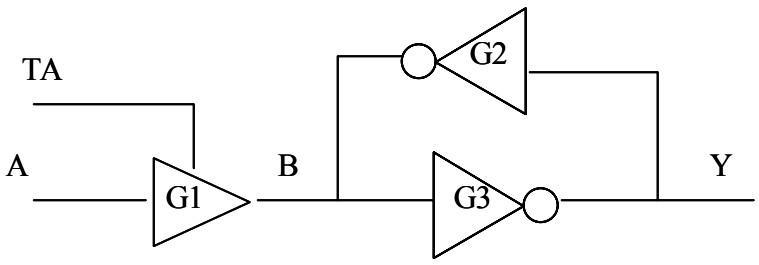
1. (10 pts) Write a VHDL process that determines if the high pulse width of signal *CLK* is less than MIN_PW_HIGH (*CLK* is std_logic type).

```
process(clk)
  begin
       if (clk = '0') then
         assert (clk'delayed'last_event >= MIN_PW_HIGH))
           report "High Pulse width violation."
           severity error;
       end if;
  end process;
```

2. (10 pts)Write a VHDL process that checks if the hold time for signal *D* is less than MIN_HOLD_TIME on the falling edge of signal *CLK*. Both *CLK* and *D* are *std_logic* types.

```
process (D)          -- must trigger on D events
 begin
    if (clk = '0') then
       assert (clk'last_event >= MIN_HOLD_TIME)
         report "Hold time violation."
           severity error;
       end if;
  end process;
```

3. (10 pts) Use a set of concurrent statements and *std_logic* drive strengths to model the following diagram:



All signals are std_logic type.

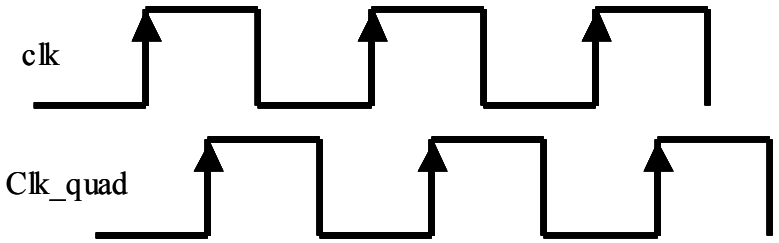
G1: When tristate control is '1', then output follows input else high impedance output.

G2 is a weak drive inverter and can be overdriven by G1.

All gate delays are 2 ns.

```
Signal  B : std_logic := 'Z';

---- G1 driver
B <= A after 2 ns when (TA = '1') else 'Z' after 2 ns;

--- G3  driver
Y <= not (B) after 2 ns;

--- G2 driver
B <= 'H'after 2 ns when (Y = '0' or Y = 'L') else
        'L' after 2 ns;
```

4. (10 pts) Write a single process that will generate a quadrature clock signal (clk_quad) from a clock signal called 'clk'. A quadrature clock is one that is offset from the base clock by a ¼ clock cycle (see below). The following is known: the clock duty cycle is 50% and it will not change for the duration of the simulation. However, the clock period is not known. Your quadrature clock can have 1 or 2 clock cycles of startup time before it begins operation.

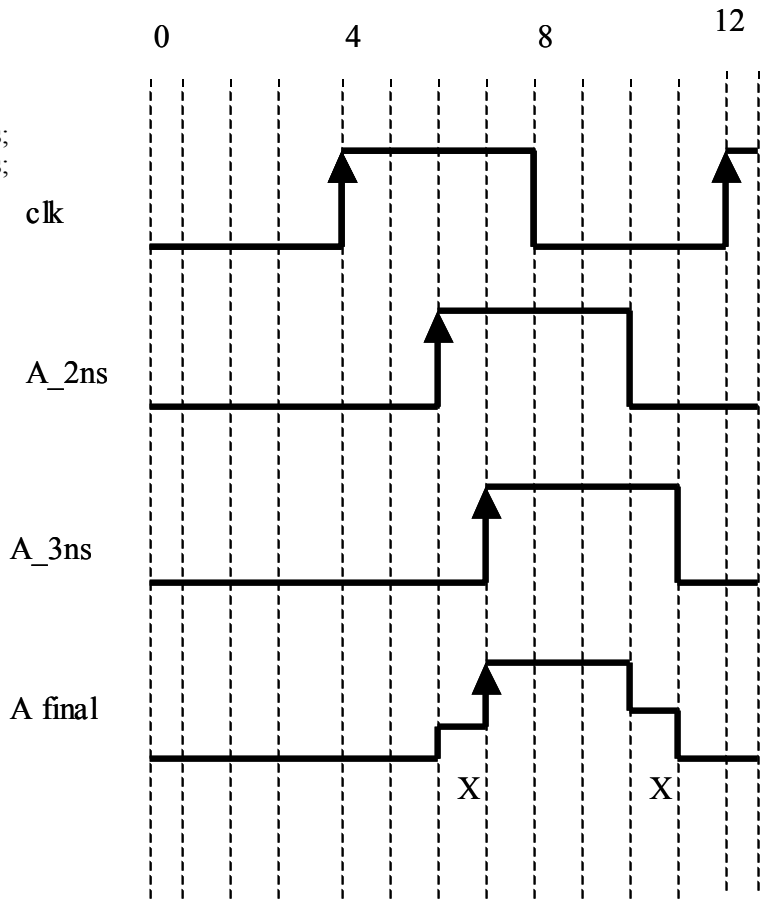



```
process (clk)
 begin
    clk_quad <= transport clk after clk'delayed'last_event/2 ;
end process;
```

5. (10 pts) Draw the 'A' waveform the results from the following code. Assume clock is a 50% duty cycle waveform with a period of 8 ns and initial value of '0'. Stop your drawing when the waveform repeats.

```
Signal  A : std_logic:=  '0';

A  <=  transport  clk after  2 ns;
A  <=  transport  clk  after  3 ns;
```

A has two drivers, so must resolve the two drivers.

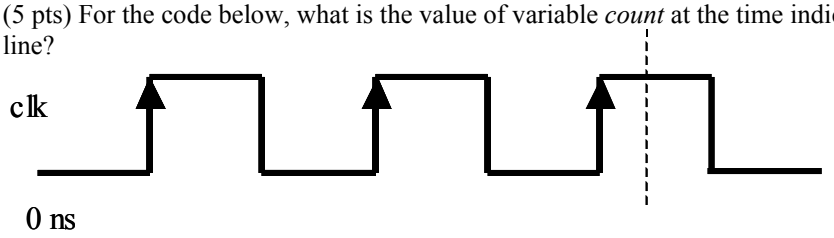


6. (5 pts) For the following code, give the final variable of shared variable 'test'. Explain your answer:

```
shared variable test: integer :=  0;

process
 begin
    test := 5;
 end process;


process
  begin
    test := 10;
  end process;
```

There are multiple problems with this code. First, neither process has a wait statement so either process will cause an infinite loop. Which process is executed first is simulator dependent, so you cannot predict what the value of *test* will be.

7. (5 pts) For the code below, what is the value of variable *count* at the time indicated by the dotted line?

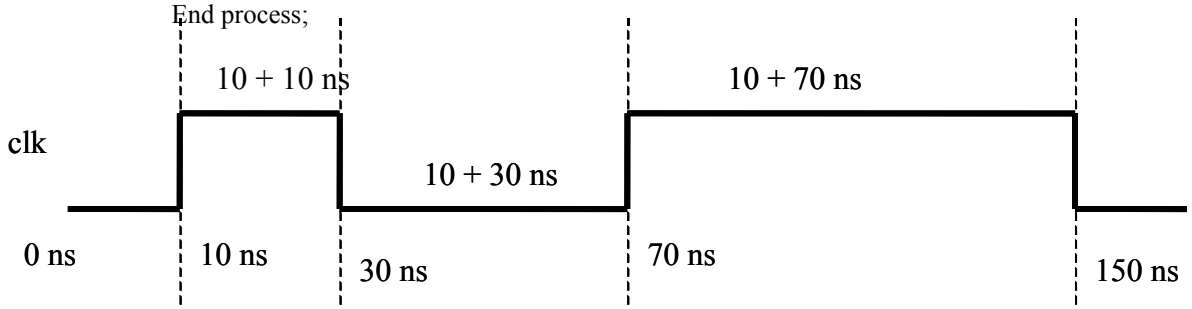



```
Process (clk)
  Variable  count : integer:= 0;

   Begin
        Count := count + 1;
   End process;
```

The final value will be 6. The process is executed at time = 0 ns so count is incremented at that point (= 1), then incremented for each event (each clock edge) for a final value of 6.

8. (10 pts) A student from Ole Miss wrote the following clock generator. Draw the waveform that is generated for two clock cycles.

```
    Signal  clk: std_logic := '0';

    Process
       Clk <= '0';
       Wait for (now + 10 ns);
       Clk  <= '1';
       Wait for (now + 10 ns);
     End process;
```



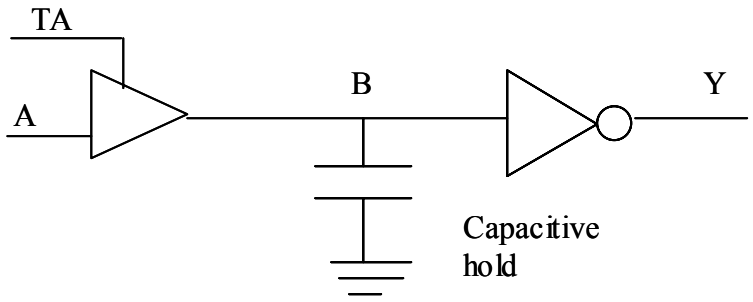
9. (10 pts) The PLD model we studied read a JEDEC file that determined the operation of the model. When was the PLD file read? Compilation, Simulation or Elaboration time? What was the advantage of writing the model in this manner?

Elaboration time – the model memory resources and execution time was proportional to the amount of resources actually used in the PLD.

10. (10 pts) Write a process that will open a file called 'output.dat' and records each time that a *std_logic* signal 'a' changes to an 'X' value from any other value and the duration of time that it remains an 'X'. Assume that the 'a' signal can only have the values '0', '1', or 'X'. The format of each line is:

```
Previous_value   Time_of_change_to_X    duration

 File   fp: text

 Process  (a)
  Variable  ll: line;
  Variable init: Boolean;

    begin
      if (init = FALSE) then
          file_open(fp, string'("output.dat"),WRITE_MODE);
         init = TRUE:
      end if;
     if (a = 'X') then
         write(ll,a);          -- write old value
         write(ll,now);      -- time of change to X
    end if;
   if ((a='0' or a='1') and (a'last_value = 'X') then
       write(ll,a'delayed'last_event);  -- write duration of 'X' value
       writeln(fp,ll);
       deallocate(ll);
  end if;
end process;
```

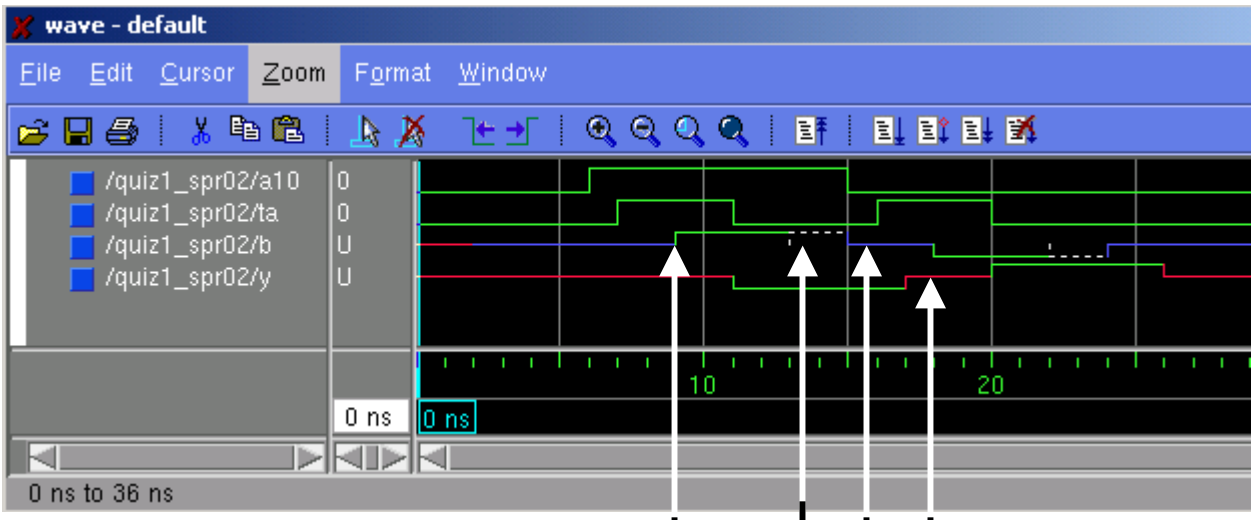
11. (10 pts) For the system below, model the Tri-state buffer and NOT gates as single concurrent statements and the capacitive hold as a single process. The capacitive hold will output either a 'L' or 'H' to hold the previous driven '0' or '1' provided by the tri-state buffer. If the tri-state buffer remains off for DECAY_TIME, the signal B should float to a 'Z' value.



When tristate control is '1', then output follows input else high impedance output. All gate delays 2 ns.

```
Signal     B : std_logic := 'Z';
Signal     A, TA, Y :std_logic;


    --tri state buffer

    B <= A after 2 ns when (TA = '1') else 'Z' after 2 ns;
    -- inverter
    Y <= not (B) after 2 ns;

    Process (B)    -- capacitive hold process
     Variable init: boolean;
      Begin
          If (init = FALSE) then
              B <= 'Z';  initial drive value for this process
             Init = TRUE;
          End if;
          If (B'last_value = '1') then
             B <= 'H',  'Z' after DECAY_TIME;
         End if;
         If (B'last_value  = '0') then
            B <= 'L', 'Z' after DECAY_TIME;
        End if;
      End process;
```

Problem #11 Simulation



1. B driven ‘1’ by TSB
2. TSB off, ‘1’ becomes ‘H’
3. After decay_time, ‘H’ becomes ‘Z’
4. Y becomes ‘X’ because of ‘Z’ input.