

Name: _____

- ```

library IEEE;
use IEEE.std_logic_1164.all;

entity PAR_TO_SER is
port (START, SHCLK: in STD_LOGIC; PAR_IN: in STD_LOGIC_VECTOR(7 downto 0);
 SO: out STD_LOGIC);
end PAR_TO_SER;

architecture ALG1 of PAR_TO_SER is
begin
 P1: process (START, SHCLK)
 variable COUNT: INTEGER := 0;
 variable DONE: BOOLEAN;
 begin
 if START = '1' then
 COUNT := 7;
 DONE := FALSE;
 elsif SHCLK'event and SHCLK = '1' then
 if DONE = FALSE then
 SO <= PAR_IN(COUNT);
 COUNT := COUNT - 1;
 end if;
 if COUNT < 0 then
 DONE := TRUE;
 else
 DONE := FALSE;
 end if;
 end if;
 end process;
end ALG1;

```

3. (15 points) For the following VHDL architecture, give the values of A, B, C, D, and E, each time a change occurs. Carry this out until no further change occurs.

```

entity prob is
 port (D : inout bit);
end prob;

architecture PROB of PROB is
 signal A, B, C, E : bit;
begin
 P1: process (A, C)
 begin
 B <= A after 3 ns;
 E <= C after 5 ns;
 end process P1;
 C1: C <= A after 10 ns;
 C2: D <= ((A or B) and C)
 or E after 1 ns;

 process
 begin process
 A <= '1' after 5 ns,
 '0' after 20 ns;
 wait;
 end process;
end PROB;

```

| Time | A | B | C | D | E |
|------|---|---|---|---|---|
| 0 ns | 0 | 0 | 0 | 0 | 0 |
|      |   |   |   |   |   |
|      |   |   |   |   |   |
|      |   |   |   |   |   |
|      |   |   |   |   |   |
|      |   |   |   |   |   |
|      |   |   |   |   |   |
|      |   |   |   |   |   |
|      |   |   |   |   |   |
|      |   |   |   |   |   |
|      |   |   |   |   |   |
|      |   |   |   |   |   |
|      |   |   |   |   |   |

Time   Event   Processes Triggered   Scheduled Transactions   Event?

| Scheduling Rules    | Transport          | Inertial                                                                                                                                               |
|---------------------|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| New before existing | Overwrite existing | Overwrite existing                                                                                                                                     |
| New after existing  | Append new         | If $v_{\text{new}} = v_{\text{existing}}$ , append new<br>Elself $t_{\text{new}} - t_{\text{existing}} >$ reject append new<br>Else overwrite existing |

4. (10 points) Develop an entity and architecture for an inhibited toggle flip-flop. This flip-flop has inputs I0, T, and Reset, and outputs Q and QN. Reset is active high and overrides the action of the other inputs. The flip-flop works as follows. If I0 = '1', the flip-flop changes state on the rising edge of T, if I0 = '0', no state change occurs (except on reset). Make the propagation delays from T to output and reset to output generics.

5. (1 point) \_\_\_\_\_ files provide realistic delays for post-layout simulation.
6. (1 point) \_\_\_\_\_ and \_\_\_\_\_ are the two main FPGA vendors.
7. (1 point) \_\_\_\_\_ (True or False) Gate delay is by the biggest contributor to circuit delay.
8. (1 point) \_\_\_\_\_ (True or False) A synthesis tool will ignore all **after** clauses in a VHDL model.

9. (5 points) If the NRE costs for CBIC and ASIC circuits are \$100,000 and \$750,000, respectively, and the cost of individual parts for CBIC and ASIC circuits are \$20 and \$6, respectively, what is the break-even manufacturing volume for these two types of circuits?
10. (10 points) Write a VHDL procedure called `scan_results` with an **in**-mode `std_logic_vector` signal parameter `results`, and **out**-mode variable parameters `majority_value` of type `std_logic`, `majority_count` of type `natural` and `tie` of type `Boolean`. The procedure counts the occurrences of '0' and '1' values in `results`. It sets `majority_value` to the most frequently occurring value, `majority_count` to the number of occurrences and `tie` to true if there are an equal number of occurrences of '0' and '1'. If the number of occurrences is equal, the procedure sets `majority_value` to 'X' and `majority_count` to the number of occurrences.

11. (15 points) Design an FSM circuit for controlling a simple home security system. The operation of the system is as follows.

Inputs:            Front gate switch (FS)  
                     Motion detector switch (MS)  
                     Asynchronous reset switch (R)  
                     Clear switch (C)

Outputs:           Front gate melody (FM)  
                     Motion detector melody (MM)

- When the reset switch (R) is asserted, the FSM goes to the initialization state (S\_init) immediately.
- From state S\_init, the FSM unconditionally goes to the wait state (S\_wait).
- From state S\_wait, the FSM waits for one of the four switches to be activated. All the switches are active-high, so when a switch is pressed or activated, it sends out a 1. The following actions are taken when a switch is pressed:
  - When FS is pressed, the FSM goes to state S\_front. In state S\_front, the front gate melody is turned on by setting FM = 1. The FSM remains in state S\_front until the clear switch is pressed. Once the clear switch is pressed, the FSM goes back to S\_wait.
  - When MS is activated, the FSM goes to state S\_motion. In state S\_motion, MM is turned on with a 1. MM will remain on for two more clock periods and then the FSM will go back to S\_wait.
  - From any state, as soon as the reset switch is pressed, the FSM immediately goes back to state S\_init.
  - Pressing the clear switch only affects the FSM when it is in state S\_front. The clear switch had no effect on the FSM when it is in any other state.
  - Any unused state encoding will have S\_init as their next state.

12. (1 point) The starting point from which a verification plan can be created is a  
\_\_\_\_\_.

13 (1 point) List one type of coverage considered during the verification process.  
\_\_\_\_\_

14. (2 points) List two layers that may be present in a layered testbench:  
\_\_\_\_\_  
\_\_\_\_\_

15. (1 point) A(n) \_\_\_\_\_ is a construct in System Verilog that represents a bundle of wires and has intelligence.

16. (10 points) Create a VHDL entity named `en_mux_821` that represents a 8:1 multiplexer with an active-low enable input which has an architecture that uses a case statement to represent the functionality of the multiplexer. Create a second entity and its accompanying architecture that represents a 16:1 decoder by using two instances of the `en_mux_821` entity.

17. (1 point) \_\_\_\_\_ (True or False) A signal initialized in its declaration will be properly initialized post-synthesis.

18. (1 point) \_\_\_\_\_ (True or False) Wait statements can be used to imply clocked behavior to the synthesis tool.

19. (1 point) The @ construct in System Verilog is similar to the \_\_\_\_\_ construct in VHDL.

20. (1 point) A \_\_\_\_\_ test environment allows you to run hundreds of tests without having to hand check the results.

21. (10 points) Develop a functional model of a 4-bit carry-look-ahead adder. The adder has two 4-bit data inputs a (3 downto 0) and b (3 downto 0); a 4-bit data output, s (3 downto 0); a carry input, c\_in; a carry output, c\_out; a carry generate output, g, and a carry propagate output, p. The adder is described by the logic equations:

$$S_i = A_i \oplus B_i \oplus C_{i-1}$$

$$G_i = A_i B_i$$

$$P_i = A_i \oplus B_i$$

$$C_i = G_i + P_i C_{i-1}$$

$$G = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0$$

$$P = P_3 P_2 P_1 P_0$$

where the  $G_i$  are the intermediate carry generate signals, the  $P_i$  are the intermediate carry propagate signals and the  $C_i$  are the intermediate carry signals.  $C_{-1}$  is c\_in and  $C_3$  is c\_out. Use a loop construct for  $S_i$ ,  $G_i$ ,  $P_i$  and  $C_i$ .



Bonus:

(5 points) A synchronous (4-bit) up/down decade counter with outputs COUNT and CO works as follows:  
All state changes occur on the rising edge of the CLK input, except the asynchronous clear (CLR).  
When CLR = 0, the counter is reset regardless of the values of the other inputs. The inputs are DATA, CLR, LOAD, EN, UP.

If the LOAD input is 0, the data input D is loaded into the counter.

If LOAD = EN = UP = 1, the counter is incremented.

If LOAD = EN = 1, and UP = 0, the counter is decremented.

If LOAD = EN = UP = 1, the carry output (CO) = 1 when the counter is in state 9.

If LOAD = EN = 1 and UP = 0, the carry output (CO) = 1 when the counter is in state 0.

Create a SystemVerilog interface for this circuit that has a clocking block that defines directions relative to the test bench.