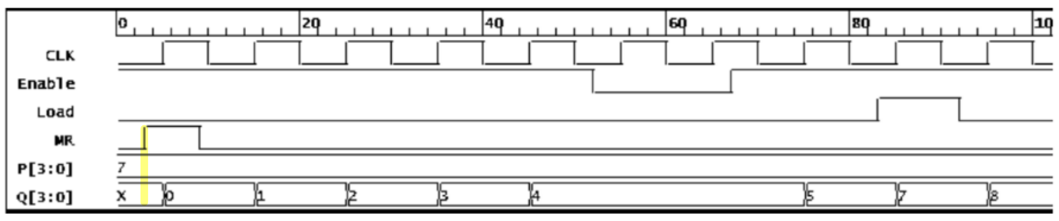
Digital Systems Testing and Verification

The Exam Consists of **Five** Questions in **Four** Pages. ***Maximum Marks: 60 Marks***

Question 1:

A. Identify the bug. * **The master reset is NOT asynchronous as expected**



B. Modify the code to correct the bug.

```
module decade_counter(P,Load,Enable,MR,CLK,Q);

  input P,Load,Enable,MR,CLK;
  output Q;

  bit [3:0] P,Q;
  bit Load,Enable,MR,CLK;

  always @(posedge CLK) begin
    if (MR)
      Q = 4'b0000;
    else if(Load)
      Q = P;
    else if (Enable)
      Q = (Q+1) % 10;
  end

endmodule
```

always @(posedge CLK or posedge MR)

C. System Verilog assertions are used to reveal errors without the need for investigating waveforms. Write 3 concurrent assertions to ensure the correct implementation of the counter specifications. [9 marks]

```
reset:   assert property ( @(posedge CLK) MR |-> (Q == 4'b0000));

load:    assert property ( @(posedge CLK) disable iff (MR) Load |-> (Q == P) );

enable:  assert property ( @(posedge CLK) disable iff (MR) (Enable && !Load) |-> (Q == (($past(Q)+1)%10)));
```

Question 2:

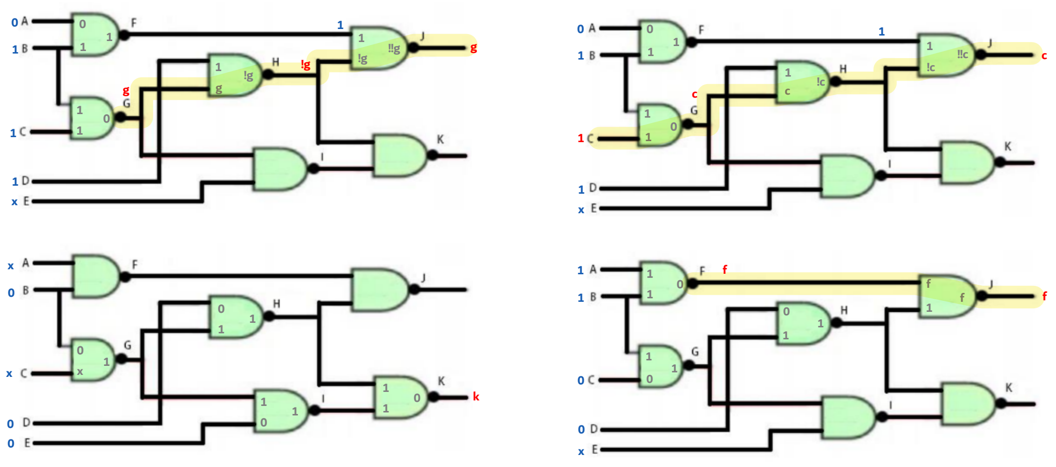
Write a coverage group to measure the coverage when using random testing. **<u>You are NOT asked to write the processor model</u>**. You are only asked to write the <u>coverage group</u>, illustrating the used <u>cover points</u>. Specify the appropriate <u>bins</u>, and the <u>illegal bins</u>. Use whatever <u>options</u> in your cover group.

```
covergroup processor @(posedge clk);
  CP1: coverpoint r_w {
         bins w = {0};
         bins r = {1};
       }
  CP2: coverpoint data {
         bins zero = {0};
         bins bin1 = {[1:64]};
         bins bin2 = {[65:128]};
         bins bin3 = {[129:254]};
         bins max  = {255};
       }
  CP2: coverpoint opcode {
         bins ops[] = {[1:9]};
         illegal_bins = default;
       }
  CP3: coverpoint address {option.auto_max_bin = 128;}
endgroup
```

Question 3: Assume they're NAND gates

A. Generate test patterns capable of detecting the following faults:

- G stcuk at 1 **->** A = 0, B = 1, C = 1, D = 1, E = x (0111x)
- C stuck at 0 **->** A = 0, B = 1, C = 1, D = 1, E = x (0111x)
- K stuck at 1 **->** A = x, B = 0, C = x, D = 0, E = 0 (x0x00)
- F stuck at 1 **->** A = 1, B = 1, C = 0, D = 0, E = x (1100x)



B. What is the difference between testing and verification.

-> Verification is the process of checking the design functionality and assure it meets the specs

-> Testing is the process of checking the fabricated chips and look for defets like stuck-at

C. State the reasonable values for the percentage of effort spent in design and in verification, commonly used in industry.

-> **70% of the design effort goes behind verification**

(In industry, a reasonable allocation of effort typically involves spending about 30-40% on design and 60-70% on verification.)

D. What is the verification technique most commonly used in the industry?

-> **Simulation based verification**

E. What is meant by Re-spin in the context of IC design?

-> **After an ASIC is finalized into a chip it fails with real hardware. The bug has to be analyzed, fixed, and the ASIC has to go through a respin, meaning getting built again.**

G. State the names of 3 verification methodologies other than UVM

(1) Simulation based verification

(2) Formal verification

(3) Semi-formal verification

(4) Assertions

F. What is meant by corner cases? Give an example.

-> **Rare events that discovered by intensive verification, example: when input values are at their maximum and minimum limits simultaneously.**

Question 4:

A. Given the following assertion:

assert property (@(posedge clk) Req |-> ##[1:2] Ack);

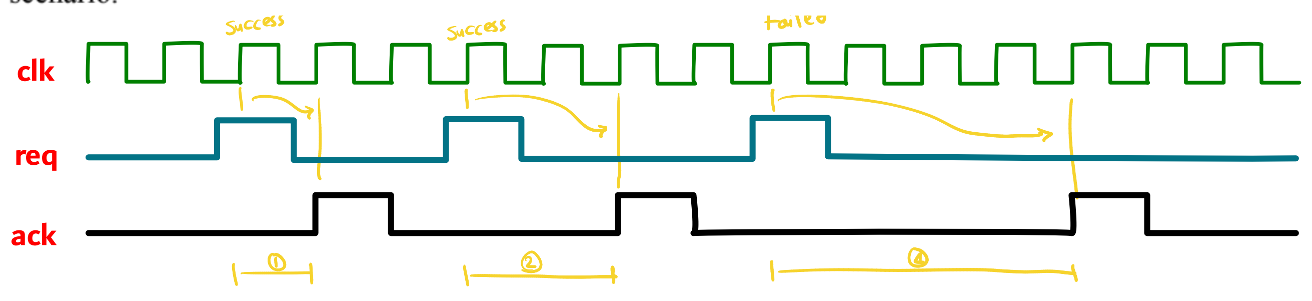
- What is the type of the above assertion?

  Concurrent

- What is the type of the implication operator?

  Overlapping

- Draw a wave diagram showing *clk*, *Req* and *Ack* in two succeeding scenarios, and one failing scenario.



B. What is wrong with the following assertion?

assert (P |-> Q);

① `assert (P -> Q);`

or

② `assert property (@(posedge clk) P |-> Q);`

C. The following code has a problem.
Determine the problem and write a modified code to solve this problem.

```
logic [4:0] addr;
logic [3:0] data;

covergroup c_group;
  cp0: coverpoint (addr + data);
endgroup
```

Problem (1): **Adding 4 bits to 5 bits, SystemVerilog considers the width of the result to be 5 bits. So only 32 bins will be generated, which's a problem as 31+15 won't find a pin because it needs 6-bits**

Solution (1): **Adding a dummy 6 bits of zeroes to the expression**

```
covergroup c_group;
  cp0: coverpoint (add + data + 6'b000000);
endgroup
```

Problem (2): **100% coverage won't be reached as the maximum result of the summation will be 46. numbers from 47 to 63 will never be covered**

Solution (2): **Include only the range from 0 -> 46**

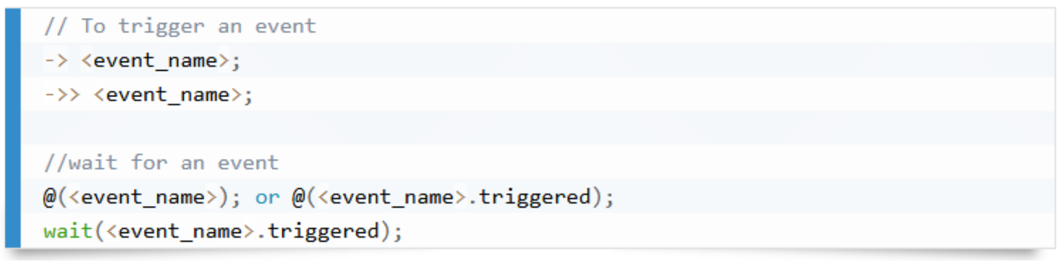
```
covergroup c_group;
  cp0: coverpoint (add + data + 6'b000000) {
    bins test_bins[] = {[0:46]};
  }
endgroup
```

Question 5:

Choose the right answer

1. An event is triggered by symbol

   a. => b. ---> c. @ [✓ d. None]



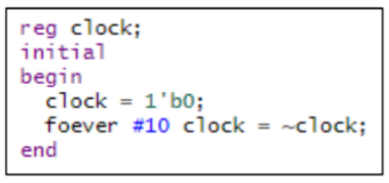
```
// To trigger an event
-> <event_name>;
->> <event_name>;

//wait for an event
@(<event_name>); or @(<event_name>.triggered);
wait(<event_name>.triggered);
```

2. Which construct is used to execute a loop for a fixed number of times?

   [✓ a. repeat] b. while c. forever d. None

3. Analyze the code below and choose the right answer



```
reg clock;
initial
begin
  clock = 1'b0;
  foever #10 clock = ~clock;
end
```

   a. The forever loop will execute the statement infinitely without advancing simulation time.
   [✓ b.] Forever loop will execute the statement infinitely and rest of the design is executed in simulation time
   c. Timing control construct is optional in forever
   d. None

4. Verilog functions have

   a. Input, output and inout port as argument
   b. Have one input argument only
   [✓ c. Can have more than one input arguments]
   d. Can have both input or output as argument.

5. Verilog functions have

   a. Registers b. Time variables [✓ c. Local variables] d. Wires

6. Which is true about task and function

   a. Task and function contain always and initial blocks
   [✓ b.] Task and function do not contain always and initial blocks and called from initial and always blocks or other tasks
   c. Task and function do not contain behavioral statement
   d. Done

7. Random number is generated by

   [✓ a. $random] b. $rand c. $strobe d. None

1111 0101 00

8. If A=1'b1, B=2'b01, C=2'b00, then y = {4{A}, 2{B}, C} equals

   [✓ a. 10'b1111010100] b. 9'b111101010 c. 8'b11110100 d. None

9. <= is used in

   a. Blocking [✓ b. Non Blocking] c. Both d. None

10. Asynchronous reset is

    a. Clock dependent [✓ b. Clock independent] c. Either d. None

---

END of Exam, Good Luck