This paper contains 7 questions

Answer ALL questions in Section A and ONLY ONE question in Section B.

An outline marking scheme is shown in brackets to the right of each question.

Section A carries 2/3 of the total marks for the exam paper and you should aim to spend about 80 minutes on it.
Section B carries 1/3 of the total marks for the exam paper and you should aim to spend about 40 minutes on it.

University approved calculators MAY be used.

A foreign language translation dictionary (paper version) is permitted provided it contains no notes, additions or annotations.
Section A

Question A1.

Write a SystemVerilog model of the ASM chart shown in Figure 1. Your model should change state on the rising edge of a clock signal and an asynchronous, active low reset should put the model into state ‘Send’. [12 marks]

![ASM Chart](image)

**Figure 1**
Indicative Solution for Question A1.

module fig1 (output logic ready,
            input logic clk, n_reset, done, valid, empty);

enum {Send, Load, Data} present_state, next_state;

always_ff @(posedge clk, negedge n_reset)
  if (~n_reset)
    present_state <= Send;
  else
    present_state <= next_state;

always_comb
  begin
    ready = '0;
    next_state = present_state;
    case (present_state)
      Send: if (Done)
        next_state = Load;
      Load: if (valid)
        begin
          ready = '1;
          next_state = Data;
        end
      else
        next_state = Send;
      Data: if (Empty)
        next_state = Load;
      endcase
  end
endmodule
Question A2.

You must show all your working to obtain full marks for this question.

Derive a test for B/0 for the circuit shown in Figure 2. What other faults are covered by this test?

Hence, derive a test for B/1. What other faults are covered by this test?
Indicative Solution for Question A2.

Test for $B/0 \Rightarrow B=1$
To transmit to $F$, $D=0$
To transmit to $H \Rightarrow E=1$
To transmit to $Y \Rightarrow G=0$
$E=1 \Rightarrow A=0$ and $C=0$
$G=0 \Rightarrow A=1$ or $C=0$ or $D=1$
So a test is $0100/1$
Also covers $Y/0$, $H/0$, $F/0$, $E/0$, $\bar{C}/0$, $\bar{A}/0$, $A/1$ Note that $C$ is transmitted by 2 paths: $C/1$ cancels itself

Test for $B/1 \Rightarrow B=0$
Hence, a test is $0000/0$, by analogy
Also covers $Y/1$, $H/1$, $F/1$, $G/1$, $D/1$, $C/1$
Question A3.

What hardware will be synthesised from the SystemVerilog module shown below?

```verilog
module thing (output logic y1, y2, y3,
               input logic d, e, c);

always_ff @(posedge c)
    if (~e)
        y1 <= '1;
    else
        y1 <= d;

always_ff @(posedge c, negedge e)
    if (~e)
        y2 <= '1;
    else
        y2 <= d;

always_comb
    if (~e)
        y3 = '1;
    else
        y3 = d;

endmodule
```

[12 marks]
Indicative Solution for Question A3.

The first always_ff block will create a flip-flop with an active low synchronous set.

The second always_ff block will create a flip-flop with an active low asynchronous set.

The always_comb block will create a combinational logic block to implement $y_3 = \bar{e} + d$ (or $e = \rightarrow d$).
Question A4.

The following SystemVerilog assertion is intended to determine whether a counter is working correctly.

```
assert property (@(posedge clk)
    load |=> ##[1:$] ready);
```

Explain the symbols used. Explain what this assertion tests and why it might be considered a poorly designed assertion

[12 marks]
Indicative Solution for Question A4.

@ (posedge clock) means that the property is tested on a rising edge.

| => is a non-overlapping implication (is tested on the next clock cycle).

##[1:$] means between 1 and infinite number of clock cycles.

The assertion states that if the load signal is asserted, the ready signal will become true eventually - i.e. an infinite number of clock cycles can pass before ready is asserted. This is poorly designed assertion because it may never complete and hence it can never fail. In other words, it tells us nothing.
Question A5.

Figure 3 shows a simple state machine. Show how a scan path can be inserted. How would the scan path be used to test whether node X is stuck at 0?
Indicative Solution for Question A5.

To create a scan path, insert a multiplexer at the input of each flip-flop, controlled by a mode signal, M (M=0, normal mode, M=1, scan mode, for example). Can reuse A as SDI and P as SDO. P+ = Q in scan mode.

To test for X/0 needs X=1, so A=0, P=1, Q=0.

Step 1. Set M=1, scan 10 into scan path (2 clock cycles). Step 2. Set M=0, A=0. Clock once to load X into Q. Step 3. Set M=1, clock once to load Q into P and hence SDO. Observe.
Section B

Question B1.

(a) What features distinguish a SystemVerilog “testbench” from other SystemVerilog models?

(b) The following piece of SystemVerilog describes a register connected to a bus.

```verilog
module busreg
    #(
        parameter N
    )
    (
        inout wire [N-1:0] sysbus,
        input logic clock, bus_control, load);

    logic [N-1:0] data;

    assign sysbus = bus_control ? data : 'z;

    always_ff @(posedge clock)
        if (load)
            data <= sysbus;

endmodule
```

Sketch a diagram of the circuit represented by this model. Write a SystemVerilog testbench for this model. Your testbench should verify that the register can store data and write it back to the bus. Your testbench should also verify that the register can be disconnected from the bus.

(c) Extend your testbench to model two such registers connected to the same bus with separate control signals. Show how the testbench can
be enhanced such that if the model does not behave as expected, a message is displayed.

[8 marks]
Indicative Solution for Question B1.

a) Testbenches have no inputs or outputs. Testbenches have initial blocks to generate inputs to the DUT.
Question B2.

(a) How are parameters defined in a SystemVerilog module? What are the advantages of using parameters? Why is it useful to give a default value to a parameter?

[6 marks]

(b) Figure 4 shows an unsigned shift and add multiplier. Registers a, b, and q have 8 bits. The adder takes two 8 bit numbers and produces a n9 bit answer (including the carry).

![Figure 4](image)

Write a SystemVerilog model of this structure. There should be one clock signal for the entire design. Generalise your model to allow the width of a, b and q to be defined by a parameter.

[10 marks]

(c) In order to use the structure in Figure 4 as a multiplier, it is necessary to load values into a and b, to clear q and to repeat the shift and add for each of the bits of b. There also needs to be a ready signal to indicate that the operation has completed. Extend your parameterised
design from part (b) to include these extra features and to include a controller.

[14 marks]
Indicative Solution for Question B2.

a) parameters are defined as:

module thing # (parameter N = 4)

The main advantage of parameterised modules is that blocks only need to be designed once. It’s a good idea to give a default value because the module can be synthesised in isolation (for debugging).

b) Something like:

module shiftadd (output logic [7:0] q, 
    input logic [7:0] a, b, 
    input logic clock);
  always_ff @(posedge clock)
    {q,b} <= {a+b,b[7:1]}
endmodule

Parameterised:

module shiftadd #(parameter N=8) (output logic [N-1:0] q, 
    input logic [N-1:0] a, b, 
    input logic clock);
  always_ff @(posedge clock)
    {q,b} <= {a+b,b[N-1:1]}
endmodule

c) The controller can simply be a counter to N.

always_ff @(posedge clk, negedge n_reset)
  if (~n_reset)
    count <= N;
  else if (count==0)
    count <= N;
  else
    count <= count - 1;

always_ff @(posedge clock)
  if (count == N)
    begin
      ready <= '1;
      q <= 0;
    end
  else
    {q,b} <= {a+b,b[N-1:1]}