This paper contains 7 questions

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

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.

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

This examination paper provides 70% of the module’s marks.

University approved calculators MAY be used.

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

(Figure 1)
Indicative Solution for Question A1.

```verilog
module fig1 (output logic ready, complete,
            input logic clk, n_reset, request, data, ack);

enum {Wait, Send, Confirm} present_state, next_state;

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

always_comb
  begin
    ready = '0;
    complete = '0;
    next_state = present_state;
    case (present_state)
      Wait: begin
        ready = '1;
        if (request)
          next_state = Send;
        end
      Send: if (data)
        begin
          complete = '1;
          next_state = Data;
        end
      Confirm: if (ack)
        next_state = Wait;
      endcase
  end
endmodule
```

Copyright 2015 © University of Southampton
Question A2.

Show that for the circuit in Figure 2 all possible stuck-at faults except one are covered by the four test vectors: 000, 011, 100, 101, where the inputs are listed in the order ABC.

Which possible stuck-at fault is not detected by any of the four vectors? Is it possible to write a test vector to detect that fault? If not, why is it not possible?
Indicative Solution for Question A2.

000 covers Y/1, D/1, E/1, F/1, A/1
011 covers Y/0, F/0, C/0, B/0
100 covers Y/0, E/0, C/0, A/0
101 covers Y/1, D/1, E/1, F/1, C/1, C/0, B/1

D/0 is not covered by these tests. No test exists because D is redundant logic. (Show this.)
Question A3.

What four pieces of hardware will be synthesised from the SystemVerilog module shown below? Which part will generate a warning, and why?

```verilog
module thing (output logic y1, y2, y3, y4,
               input logic d, e, c);

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

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

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

always_comb
    if (~e)
        y4 = y4;
    else
        y4 = d;

endmodule
```

[13 marks]
Indicative Solution for Question A3.

The first always_ff block will create a flip-flop with an active high enable.

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

The first always_comb block will create a combinational three state buffer.

The second always_comb block will create a latch with an active high enable. This block will generate a warning because an asynchronous latch has been created.
Question A4.

What hardware does the following SystemVerilog code represent? What is the expected behaviour of this code?

```systemverilog
always_ff @(posedge clock, negedge n_reset)
    if (˜n_reset)
        a = '0;
    else
        a = b;

always_ff @(posedge clock, negedge n_reset)
    if (˜n_reset)
        b = '0;
    else
        b = ˜a;
```

Explain why the simulated behaviour of this code might depend on the order in which the two blocks are evaluated. How should the code be modified to ensure that the behaviour is predictable? Why does your modification work?

[13 marks]
Indicative Solution for Question A4.

The code shows two flip-flops, connected in a ring, with an inverter. (In fact, it’s a 2-bit Johnson counter.) We would expect the sequence: 00, 01, 11, 10, 00, ...

The code is not deterministic because blocking assignments are used. This means that the ordering of events in the event list is non-deterministic, so at a clock edge, it’s not known which flip-flop is evaluated first and hence whether the value before or after the clock edge is assigned in each case. (In fact, my simulation shows that both this model and one with the flip-flops compiled in the reverse order both fail, but in different ways!)

The solution is to use nonblocking assignments. This ensures that the right hand sides are evaluated at the clock edge, but the assignments are not completed until after all blocking events have been done.
Question A5.

What does the following expression mean when used in an ASM chart?

\[ Y \leftarrow A.B \]

Draw a diagram of the hardware that would implement this expression. [10 marks]
Indicative Solution for Question A5.

This is a register transfer operation. The value of A.B is loaded into register, Y, at the end of the current clock cycle and retained until redefined.

One hardware implementation is a register with a MUX. The output from the register is fed back into one MUX input. The other MUX input is connected to the output of an AND gate.
Section B

Question B1.

(a) What are the benefits of using Assertion Based Verification? [5 marks]

(b) Design an “averager” to the following specification. On each clock cycle, a 4-bit integer, A, is loaded into the system. A is added to a stored value, S, and the average is calculated by dividing the sum by 2. The result is output from the system and also becomes the new stored value, S. There should be one clock signal and one asynchronous reset that sets S to 0.

Write a SystemVerilog model of this design. [10 marks]

(c) Write a testbench in SystemVerilog for the averager. [10 marks]

(d) Write a SystemVerilog assertion to verify that the averager works correctly. [10 marks]
Indicative Solution for Question B1.

a) The main benefit of ABV is that it removes the need to inspect complicated waveforms in order to determine whether a design works correctly.

b) Something like:

```verilog
defined module ave (output logic [3:0] s, 
    input logic [3:0] a,
    input logic clock, n_reset);

global [4:0] av;
always_ff @(posedge clock, negedge n_reset)
    if (˜n_reset)
        s <= ’0;
    else
        s <= av[4:1];

always_comb
    av = s+a;
endmodule

c)
defined module test_av;

global [3:0] s, a;
global logic clock, n_reset;
ave a0 (.*);

initial
    begin
        clock = ’0;
        n_reset = ’1;
    #5ns n_reset = ’0;
    #5ns n_reset = ’1;
        forever #10ns clock = ~clock;
    end

initial
    begin
        #25ns a = 2;
        #20ns a = 5;
    end

d)

property Check;
```

---

Copyright 2015 © University of Southampton

Page 13 of 17
@ (posedge clock) a |=> (s == (s+a)/2);
endproperty

assert property (Check);
Question B2.

(a) Explain the principle of Scan-In, Scan-Out (SISO).  

[5 marks]

(b) Figure 3 shows a simple state machine, with two state variables, S and T, a single input, A, and a single output, S.

Consider just the combinational part of the circuit (shown within the dashed lines). Write a test for M/1 (L stuck-at 1), in terms of S, T and A. What is the value of $T^+$ if the fault, M/1, does not exist? What is the value of $T^+$ if the fault does exist? What other faults in the combinational part of the circuit are covered by this test pattern? (Note that $\overline{S}$ and $\overline{T}$ must be the inverse of S and T, respectively.)

[10 marks]

(c) Explain how a scan-path can be included within the circuit of Figure

TURN OVER
3. Explain how the test that you derived in part (b) can be applied to the full, sequential circuit. What would be observed at the output of the scan-path if the fault M/1 does not exist, and also if it does exist? [10 marks]

(d) How would you test the scan-path, from part (c), for stuck-at faults? What is the consequence of a stuck-at fault in the scan-path? [10 marks]
Indicative Solution for Question B2.

a) The principle of SISO is to insert a multiplexer at the input of each flip-flop in a sequential circuit and to connect one input of each mux to the output of the preceding FF, such that the FFs may be configured as a shift register for testing purposes. This allows the combinational part of the circuit to be tested as if it were separated from the flip-flops.

b) To test for M/1 implies that M=0. Therefore S=T=1. To transmit the value to T requires that L = 1. Therefore A or S or both must be 0. As S is already 1, S is 0 and therefore A is don’t care.

In the fault-free case, T is 1, if faulty, 0.

This test will also cover T, S, T/0. (But not L/0.)

Note also that the path through S is sensitised. S = A. We have two cases:

- A=1 covers S/0, J/1, T/0.
- A=0 covers S/1, J/0, K/0, A/1, T/1.

c) There are several ways to insert a scan path. Let’s assume that there is a control signal, B, which enables the scan path when 1. Assume, in scan mode, A is connected to the scan i/p of T; T is connected to S and the scan data output is S. (This is the easiest way to do it!)

So, to set S and T to 1, we set B to 1, A to 1 and clock twice.

Then, set A to 1 (or 0), set B to 0, clock once. This loads T and S into the scan chain. We can observe S at this point and observe any errors in that part of the combinational logic.

Set B to 1, clock once. This moves T to S. We can now observe the value at S, so if M/1 does not exist, S is 1, otherwise S is 0.

d) The scan-path needs to be tested by applying a sequence, such as 0101 through the scan-path, when it is in scan-mode. If a stuck fault exists, only 0s or 1s will be observed at the scan output. Hence the circuit will be identified as faulty and rejected, even if the rest of the circuit functions correctly.

END OF PAPER