2013 Paper E2.1: Digital Electronics II

Answer ALL questions.

There are THREE questions on the paper.

Question ONE counts for 40% of the marks, other questions 30%

Time allowed: 2 hours

(Not to be removed from the Examination Room)

Information for Candidates:

The following notation is used in this paper:

1. Unless explicitly indicated otherwise, digital circuits are drawn with their inputs on the left and their outputs on the right.
2. Within a circuit, signals with the same name are connected together even if no connection is shown explicitly.
3. The notation X2:0 denotes the three-bit number X2, X1 and X0. The least significant bit of a binary number is always designated bit 0.
4. Signed binary numbers use 2's complement notation.

1. (a) *Figure 1.1* shows a counter circuit where one of the flip-flops has an asynchronous reset input which forces Q2 to zero whenever M=1. For M=0, draw the state diagram for the circuit of Figure 1.1 assuming that all flip-flops are initially in the reset state. Draw a timing diagram showing the waveforms of the clock CLK, and the outputs Q0, Q1 and Q2 for one complete cycle of the counter.

[6]

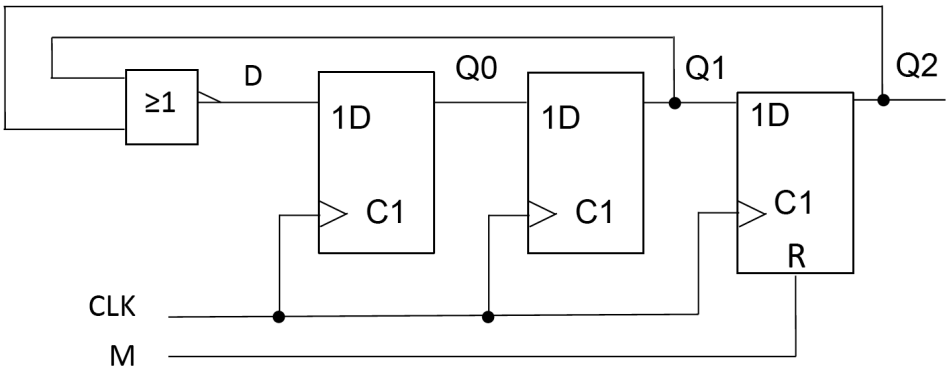


Figure 1.1

(b) Repeat (a) above, but for the case where M=1. By considering the output Q0, state the behaviour of this counter in relation to the control signal M.

[4]

(c) The circuit in *Figure 1.1* is implemented on an Altera Cyclone III FPGA. Each Logic Element (LE) of the FPGA consists of a 4-input lookup-table (LUT) with a propagation delay t_d where 1ns $\leq t_d \leq$ 3ns, and a flip-flop with clock-to-output delay t_p where 2ns $\leq t_p \leq$ 2.5ns, a setup time t_s where 2ns $\leq t_s \leq$ 3ns, and a hold time t_h of 5 ns.

(i) Derive the inequalities for the setup and hold times applied to the leftmost flip-flop.

[6]

(ii) Hence or otherwise calculate the maximum clock frequency for the circuit.

[3]

(iii) How many LEs are required to implement this circuit?

[1]

(d) Write in Verilog HDL the behavioural description of a synchronous counter circuit as shown in *Figure 1.2*, with an input S and an output TC. TC goes high for one clock cycle in every 4 cycles of the clock signal CLK if S=0, and in every 5 cycles of CLK if S=1.

[10]

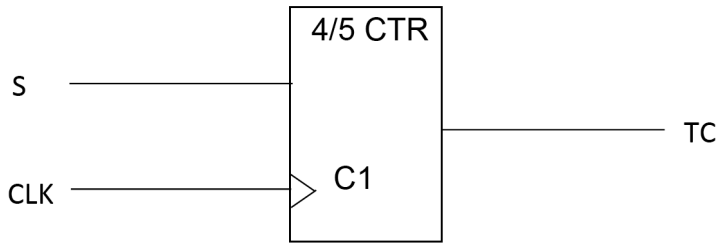


Figure 1.2

(e) X7:0 is an 8-bit signed number in 2's complement form. Determine the hexadecimal number range or ranges of values of X7:0 for which the following expressions are true:

(i) $X7 \cdot \overline{X6} \cdot (X5 + X4)$

[5]

(ii) $\overline{X7} \cdot (X6 \oplus X5)$

[5]

2. (a) Compare the advantages and disadvantages of a flash and a successive approximation A-to-D converter. [5]

(b) *Figure 2.1* shows the circuit of a 3-bit successive approximation A-to-D converter. The circuit consists of a 3-bit D-to-A converter (DAC), an analogue comparator (CMP), a NOT-gate (G), a register (REG) and a logic block (Logic). The DAC produces an analogue output value X=Q2:0–0.5 volts in the range of -4.5v to +3.5v. The comparator output HI is true whenever the input V is greater than the DAC output X. On the rising edge of the clock signal CLK, the register is loaded with the new data if START=DONE=0.

The register and logic block form a synchronous state machine whose state is defined by the signed 3-bit number Q2:0. Together they implement the successive approximation algorithm with a state diagram as shown in *Figure 2.1* when START=DONE=0. A conversion is initiated by START going high for one clock cycle to reset the register.

(i) Explain why the sequence of states that should be followed when converting an input voltage V= –1.9 volts is 0, –2, –1, –2. State the relationship between the input voltage V and the converter value Q2:0 at the end of a conversion. [5]

(ii) Draw a timing diagram showing the process of conversion when V = –1.9 volts. Your diagram should show the waveforms of CLK, START, HI, D and DONE, and give the values of X and Q0:2. [8]

(iii) Write in Verilog HDL a description of the synchronous state machine, which will synthesis all the digital circuits in Figure 2.1 (i.e. everything excluding CMP and DAC). [12]

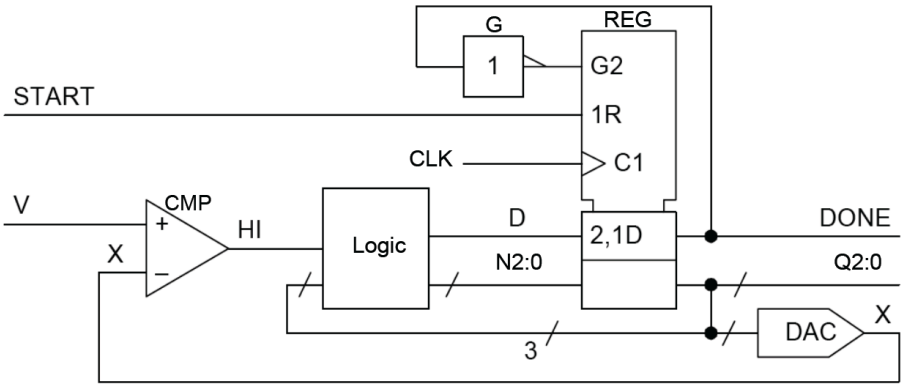


Figure 2.1

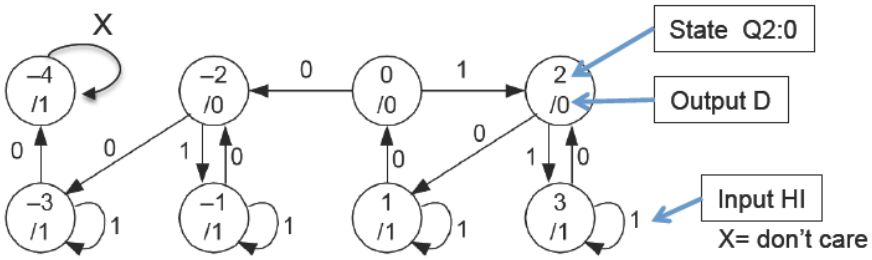


Figure 2.2

3. *Figure 3.1* depicts a 16-bit microprocessor connected to three memory circuits: one 16k×8bit read/write memories (RAM) and two 8k×8bit read-only-memory (ROMs). The address (A), data (D), write enable (WE) and output enable (OE) signals are connected to multiple memory circuits as indicated in the diagram. The symmetrical signal CLOCK has a period T.

(a) The microprocessor has a 16-bit address bus, and the starting address of the RAM and the ROM chips are, in hexadecimal, $C000, $0000 and $4000 respectively. State the address range for each memory circuit. Design, in Boolean equations and in Verilog HDL description, the "Decoder Logic" module that generates the chip-enable signals for the memory circuits.

[12]

(b) *Figure 3.2* shows the timing specifications for the microprocessor when it is reading from memory. Data is read into the microprocessor 1½T after the start of the cycle with a setup time of t_s. The propagation delays of the RAM from its address, chip enable and output enable inputs are t_A, t_C and t_O respectively. The decoder logic is implemented on a Cyclone III FPGA where each lookup table has a propagation delay of t_g. The microprocessor asserts the address signals A15:0 and the write signal WRITE t_d after rising edge of the clock, and the memory access signal MEM t_m after the falling edge of the clock.

Write down all the timing inequalities that must be satisfied in order to meet the setup time requirement t_S when reading data from the RAM.

[10]

(c) Determine the smallest clock period, T, for which the t_S requirement is satisfied if the timing parameters (in ns) are: t_g=3, t_d=8, t_m=6, t_s=4, t_A=14, t_O=4, t_C=12.

[8]

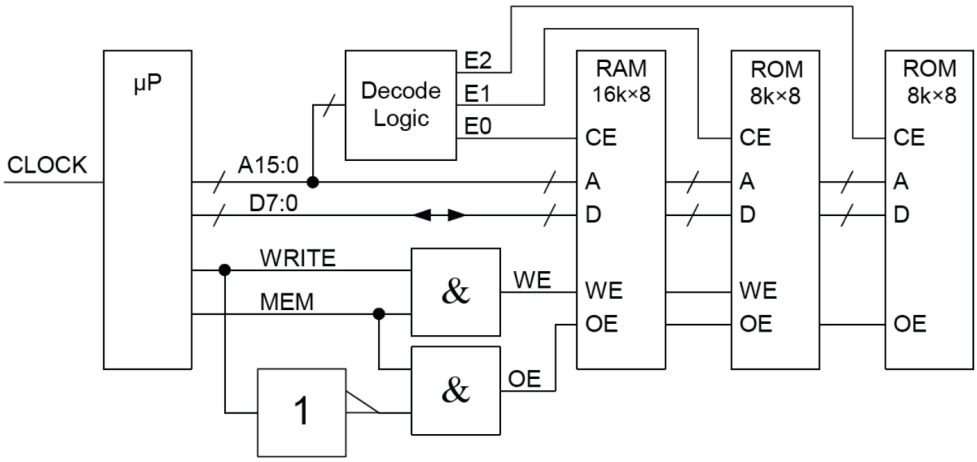


Figure 3.1

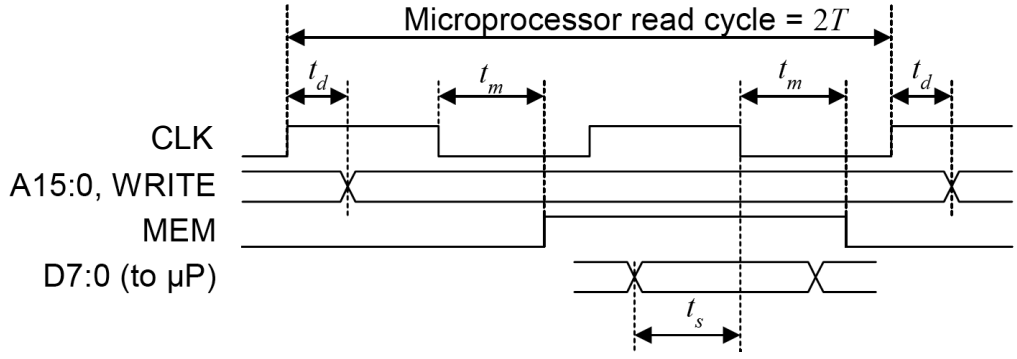


Figure 3.2

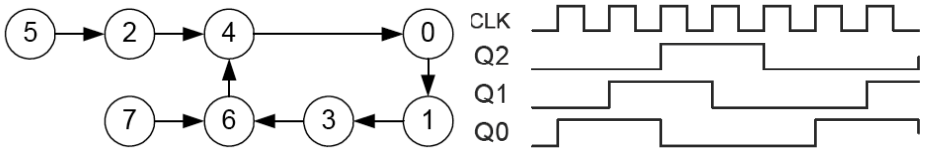
2013 Paper E2.1: Digital Electronics II- Solutions

(With annotations)

1. (a) This question tests student's ability to analyse a simple state machine. It is helpful to draw the transition table first:

Q2	Q1	Q0		Next_Q2	Next_Q1	Next_Q0 (D)
0	0	0		0	0	1
0	0	1		0	1	1
0	1	0		1	0	0
0	1	1		1	1	0
1	0	0		0	0	0
1	0	1		0	1	0
1	1	0		1	0	0
1	1	1		1	1	0

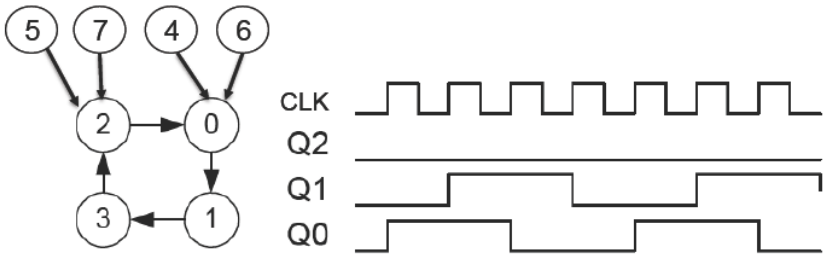
Expect to show the illegal state too!



[6]

Most students missed out the "illegal states". Some ignore the question requiring the state diagram, or state transition table. Any one of those will be acceptable. Easy question – most got this question right.

(b)



If M = 0, CLK is divided by 5 otherwise CLK is divided by 4. Therefore it is a selectable divide by 4 or 5 counter.

[4]

Only a few students managed to get the function of the circuit correct (i.e. how M affects the way the divider works). Of course, it is only worth 1 mark out of 4.

(c) This question tests student's understanding of timing constraints in digital circuits, particularly within a FPGA.

(i) Setup time constraint: $tp\_max + td\_max + ts\_max < T$
Hold time constraint: $tp\_min + td\_min > th\_max$

[6]

(ii) Setup time: 2.5 + 3 + 3 < T, therefore T > 8.5ns, or Fmax = 117.6MHz.
Hold time: 2 + 1 > 5, hold time could be violated.
This circuit has unsolvable hold time violation and therefore may not work.
If tp_max and td_max occurs, then there would be no hold time violation because 3 + 2.5 is greater than 5.

[3]

(iii) Since each LE has a LUT and a flip-flop, this needs 3 LEs.

[1]

It is surprising how many students did not get part (iii), indicating that they did not understand what is inside a LE! I think I blame myself for not getting through in my teaching.

(d) This question tests student's knowledge about using Verilog to specify hardware.

```
module  count_4_or_5 (TC, S, CLK);

input       S, CLK;          // S = 0, count 4 CLK, otherwise count 5 CLK
output      TC;              // timeout signal; goes high for 1 cycle when count is 0
reg [2:0]   count            // easier to count down; this stores the count value
reg         TC;

initial
            count = 3'b0;
            TC = 1'b0;

always @ (posedge CLK)
      if (count == 3'b0)  begin
               TC <= 1;
               If (S == 1'b1) count <= 3; else count <= 4;
            end
      else begin
               TC <= 0;
               count <= count – 1'b1;
            end
endmodule
```

[10]

It question is easy – it is really a slight modification of examples given in lectures. It should be extremely straight forward and an easy 10 marks to pick up. Counter in Verilog is so useful that everyone should be able to do at least half of this! Unfortunately, a significant proportion of students have not bordered to learn enough Verilog to provide a reason answer – disappointing.

(e)

(i) $X7 \cdot \overline{X6} \cdot (X5 + X4)$

Two ranges: 101X XXXX => 1010 0000 to 1011 1111 or $A0 to $BF

10X1 XXXX => 1001 0000 to 1011 1111 or $90 to $BF

Combined range: $90 to $BF or -112 to -65.

[5]

(ii) $\overline{X7} \cdot (X6 \oplus X5)$

Two ranges: 001X XXXX => 0010 0000 to 0011 1111 or $20 to $3F

010X XXXX => 0100 0000 to 0101 1111 or $40 to $5F

Combined range: $20 to $5F or 32 to 95.

[5]

The mention of "2's complement" representation is a bit of a "trick". Essentially bits are bits, and an 8-bit number in any format is just a just a number with 8 bits. How we interpret the number is really up to us. So, the mention of 2's complement is a "red herring". The answer could be given in either hexadecimal or signed decimal. Much easier to work in hex then signed number!

Overall, question did serve its purpose as the "Mastery" question. Most students who obtained 25/40 or above had no difficulties in passing this paper. Those scoring 35 or above would obtain at least a B, and mostly like an A. In contrast, those scoring 15-20 are mostly like to get around 40% or even fail!

For the class, the average marks of Q1 is 26.2 out of 40, which is lower than my target of 30 out of 40, but not by much.

2. This question tests student's understanding of A-to-D converter, particular the design of successive approximation ADC.

(a) Book work. Flash converters: + very fast and simple, suitable for MHz sampling rate; - complexity grows exponentially with no of bits, suitable for at most 8-bit converters; expensive. SA ADC: + reasonably fast, generally good for up to 12 bit ADC, reasonably low cost, complexity grows linearly with no of bits; - cannot do more than around 100's of ksample/sec, needed multiple clock cycles.

[5]

A handful of students did not read the question, and wasted loads of time describing how flash and SA ADC work, which of course is not what is required. Otherwise, most students earned valuable marks with this.

(b) (i)

The range of DAC is -4.5v to 3.5v for 3-bit signed number at the input, therefore the ADC would convert within this range too. The sequence of comparisons (without referring to the detail working of this circuit, but just the SA-ADC algorithm) on each clock cycle must therefore be:

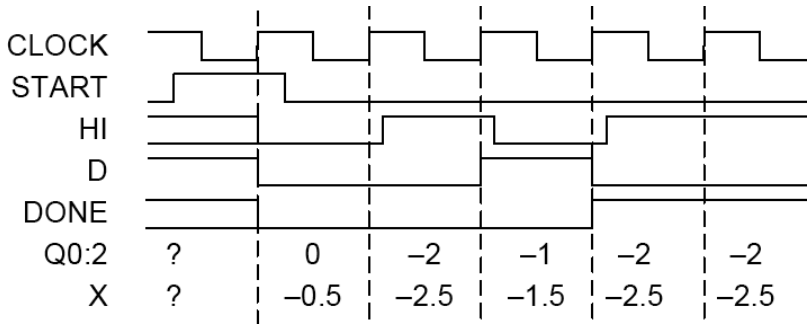
Step 1: state = 0, X=-0.5v (halfway between -4.5v and 3.5v) > -1.9v
Step 2: state = -2, -2.5v (halfway between -4.5v and -0.5v) < -1.9v
Step 3: state = -1, -1.5v (halfway between -2.5v and -0.5v) > -1.9v
Step 4: state = -2.

Q2:0 = ROUND (V).

[5]

The key is to know what the values of X are in each comparison. Only a small number of students got the relationship between Q2:0 and V.

(ii)



[8]

This part is a give away if one can do (i). It really shows if someone understood the working of SA-ADC or not!

(iii)

```
module  SAR_FSM (DONE, Q, START, HI CLK);
  input       START, CLK;     // START = 1 for 1 cycle to start a coversion
  input       HI;             // HI = 1 if input voltage is higher than DAC voltage
  output      DONE;           // Goes high at end of conversion
  output [2:0]  Q;            // Drive DAC input
  reg [2:0]   Q;              // Also used as internal states

  parameter       S_zero = 1'b000;
  parameter       S_one = 1'b001;
  parameter       S_two = 1'b010;
  parameter       S_three = 1'b011;
  parameter       S_minus_one = 1'b111;
  parameter       S_minus_two = 1'b110;
  parameter       S_minus_three = 1'b101;
  parameter       S_minuus_four = 1'b100;

  always @ (posedge CLK)
        if (START==1'b1)  begin         // start a conversion
                Q <= S_zero;
                DONE <= 1'b0; end
        else
                if (DONE == 1'b0)
                  case (Q)
                     S_zero: begin
                                DONE <= 1'b0;
                                If (HI==1'b0) Q <= S_minus_two; else (Q <= S_two);
                              end
                     S_one: begin
                                DONE <= 1'b1;
                                If (HI==1'b0) Q <= S_zero; else (Q <= S_one);
                              end
                     S_two begin
                                DONE <= 1'b0;
                                If (HI==1'b0) Q <= S_ one; else (Q <= S_three);
                              end
                     S_three: begin
                                DONE <= 1'b1;
                                If (HI==1'b0) Q <= S__two; else (Q <= S_three);
                              end
                     S_minus_one: begin
                                DONE <= 1'b1;
                                If (HI==1'b0) Q <= S_minus_two; else (Q <= S_minus_one);
                              end
                     S_minus_two: begin
                                DONE <= 1'b0;
                                If (HI==1'b0) Q <= S_minus_three; else (Q <= S_minus_one);
                              end
                     S_minus_three: begin
                                DONE <= 1'b1;
                                If (HI==1'b0) Q <= S_minus_four; else (Q <= S_minus_three);
                              end
                     S_minus_four: begin
                                DONE <= 1'b1;
                                Q <= S_minus_four;
                              end
                  endcase
  endmodule
```

[12]

This Verilog code is quite long, therefore the smart thing to do is to show only some of the states, as long as it is clear that you know how to fill in the missing blanks, you won't loose much (if any) marks. Many students DID NOT have the test for START==1 at the beginning – of course that is need to start the conversion.

Although this question is not easy, part (a) is practically a give a way, and (b) (iii) Verilog coding is also an easy part to earn some marks – as long as one is comfortable with describing a FSM in Verilog. The average marks for this question was 17 out of 30, which is close to my target of 15 out of 30.

3. (a)

RAM (16k): $C000 - $FFFF

ROM (8k): $0000 - $1FFF and $4000 to $5FFF.

E0 = A15*A14

E1 = /A15*/A14*/A13

E2 = /A15*A14*/A13

```
module  decoder (E0, E1, E2, A);

    input [15:12]    A;              // 4 MSB of address bus
    output       E0, E1, E2;         // chip enable signals

    assign  E0 = A[15] & A[14]
    assign  E1 = ~A[15] & ~A[14] & ~A[13];
    assign  E1 = ~A[15] & A[14] & ~A[13];
endmodule
```

[12]

It is surprising how many students got this wrong! Somehow, they use calculator to find out what 16,000 is in hexadecimal – of course in computing and digital terms, 1k is NOT 1000, but 1024! The trick is NOT to use calculator, but use common sense. $1000 (in hex) is 4096 or 4k – this is something I have mentioned time and time again in lectures. Then, the rest is easy!

Any other common mistake was that when one calculates 8k, and found that it is $2000. Then students just add $2000 to $4000, and say the range for ROM 2 is $4000 to $6000. Equally RAM range is $C000 to $10000!!! Of course if you count 0 to 10, you get 11 numbers, not 10! In other words, don't forget the 0.

Finally, many use always @(posedge CLK) in the decoder Verilog code – showing lack of understanding. The decoder is just a combinatorial logic; therefore it does have a clock. Others used if-else statement – this is possible, but tedious. The logic synthesis programme would simplify this if-else statement to the same Boolean logic, but as Verilog code, it is long winded and lack elegance.

(b)

There are four different paths to consider:

Path	Inequality	Actual timing
A15:0 ➔E0 ➔CE➔D	$td + tg + t_C + ts < 1½T$	$27 < 1½T$, $T > 18ns$
A15:0 ➔A➔ D	$td + t_A + ts < 1½T$	$26 < 1½T$, $T > 17.3ns$
WRITE ➔OE ➔D	$td + 2tg + t_O + ts < 1½T$	$21 < 1½T$, $T > 14ns$
MEM ➔OE ➔D	$tm + tg + t_O + ts < T$	$T > 17ns$

[10]

This part is quite hard. Many did not consider EACH of the timing, but just gave a generic setup and hold time requirement. This really tests student's full understanding of digital timing constraints.

(c) By substituting all the numerical timing values into the inequalities, we found that the critical path is the one through the decoder logic. Since the decoder logic equation for E0 only has 4 input terms, this can be implemented using a single LUT with a delay of 3ns. Therefore T > 18ns.

[8]

Overall, students did not do too well on this question – which is surprising because part (a) is worth 12 marks, and is really very easy. Part (b) and (c) are difficult, but they are there to separate the A-graders with the average! The average marks for Q3 is only 11.5 out of 30, which is lower than the target of 15 out of 30.

This paper turns out to be harder than I expected. There are 16 students (out of 152 registered, and 149 who sat the paper) who failed, which is slightly higher than the nominal 10% limit. Nevertheless, all questions (or all part of Q1) had students who scored 100%, demonstrating that all of the questions are "do-able". One thing is clear – given that one has to write three Verilog code segments, 2 hours may not have been enough. The average marks for three questions and the entire paper are:

Q1: 26.2% out of 40%

Q2: 17.2% out of 30%

Q3: 11.5% out of 30%

Overall: 54.9% (ideal range: 55% to 65%).