Comments  Please see the attached report.
Section A

Q1

Q1 and Q2 are compulsory.

A compulsory question marked out of 10, where there is some choice and students can answer 5 parts out of 8. A number of students choose to answer more than 5 questions, sometimes all 8, in this section. Whilst I am often generous and take the 'best' five marks, attempting more than is required is somewhat of a waste of time. Time is better at answering questions that will contribute to final mark, rather attempting to maximise just a few marks.

a)

The aim of this question was to fully specify the values in the required format, by specifying the number of bits and the base of the number. ½ mark for each question.
   i) you are required to specify the 8-bit binary equivalent of +25, i.e. 8'b00011001.
   ii) 16'd65 (although I did allow a binary representation)
   iii) 16'hFFEE
   iv) 8'h119

Problems:
In a number of cases the number of bits was left out, the base was not specified, or the value was not specified in the required format. In part iii) the two's complement negative value, FFEE, was not always specified, instead a negative value was quoted.

b)

The aim of this question was for you to discuss the difference, in terms of the format of the sensitivity list, between an always block containing blocking or non-blocking assignments. In the case of a block containing blocking assignments the sensitivity list would contain either all the inputs that cause a variable (or variables) in the block to be re-evaluated, or in more generally *. In the case of a block containing non-blocking assignments, then we have a sequential block, so the sensitivity list contains some timing dependency, such as posedge, or negedge clock. 1 mark for discussing each.
Problems:
A number of answers focused on discussing the difference between the way blocking
and non-blocking assignments are evaluated – the question was not asking this! In some
cases the discussion focused on discussing only one case, i.e. that in the case of non-
blocking the sensitivity list contains some dependency on the edge of a clock signal, i.e.
posedge clock (which is correct) but then preceded to say that in the case of blocking it
doesn’t! Some answers discussed blocking as representing an asynchronous circuit!

c)
The aim of this question was to spot the errors in the Verilog module provided. The
errors include:
• Module name starting with a number – tricky but a few spotted this!
• Output Q not defined as a reg – blocking statements are used
• Sensitivity list – should use * and not posedge S
• Missing endmodule

On reflection I also allowed missing begin as a error, although strictly speaking a begin
and end is not required. In a number of cases only 2 errors were reported, I suspect this
was because the question is worth 2 marks, rather than they could only find 2 errors. In
fact there were 4 errors - ½ mark each!

Problems:
Some random errors were identified, such as:
• The number of bits is not specified for input S – why?
• The inputs need to be declared as reg – why?
• The assignments should be non-blocking as there is a posedge in the
  sensitivity list – you are told that the block is combinatorial!

d)
This was a relatively well answered question. 1 mark was awarded for some evidence of
DeMorgan’s theorem being used, 1 mark for a correct answer.

Problems:
Working out $\overline{Q}$ and not Q! Not providing a product of sums expression as requested.

e)
The aim of this question was to test your understanding of the operation of a D-type
flip-flop; in this case with a CE input – which you have come across in the lab. The
question sort of gave the answer away by defining CE – clock enable - which simply
enables the clock and controls on which clock edge the flip-flop is loaded – 1 mark for a
suitable description. You were also asked to produce a sketch of a timing waveform to
show how it works – 1 mark for a sketch.
Problems:
Not adequately describing the operation of CE – there were a lot of answers that failed to even mention ‘clock’ in the discussion! Waveform sketches that do not include all the required signals (in particular CE and clock?).

f)

The aim of this question is to test your ability to produce an exhaustive test stimulus for the given design, this involves testing all possible combinations of the input signals, D0, d1 and s. The stimulus should be implemented as an initial block with structure:

```
initial
begin
  d0 = 0;
  d1 = 0;
  s = 0;  // set inputs
  #10    // add delay
  // do rest

$stop
end
```

Testing all possible input combinations – 1 mark, initial, begin, $stop and end - ½ mark, using sensible delays after setting inputs – ½ mark.

Problems:
- Creating a clock – where is a clock used in the design?
- Not producing a test stimulus that is exhaustive – the question does ask for this!
- Missing out the initial, begin … end, and $stop.

g)

The aim of this question is to test your ability to extract a Boolean expression from a truth table and then translate this to a circuit schematic. 1 mark for the Boolean expression, 1 mark for an appropriate circuit diagram. Note, the Boolean expression can be simplified – if the non-simplified expression is given then only ½ mark was awarded.

Problems:
On the whole the extraction of a Boolean expression was handled well. However, it was clear some did not know how to do this! When it came to drawing the circuit diagram, it was clear that some hadn’t read the question! You are told that standard gates are available – AND, OR and NOT, yet still some logic gates were drawn with bubbles at the input to represent inversion – you don’t have such a gate available!
h) This was a wordy question and involved discussing the operation of the datapath and control blocks in a sequential system – bookwork. 1 mark for an adequate discussion of both.

Problems:
It was surprising how many answers failed to discuss the operation (as requested) and simply drew a diagram with no discussion! Not identifying the interplay between control and datapath, i.e. the control tells the datapath what to do with the data. Not recognising that the control consists of a FSM.
Q2

A compulsory question marked out of 10. Answer 5 parts out of 8. The majority of students attempted this question even if they attempted only fewer parts or in very few cases more than five parts although they should not have done so.

a)

Standard lecture material related question. It was attempted by most of the students and the majority got it right. One mark was awarded for the right order and another one for the correct mapping of the memory hierarchy levels to memory technologies. Half marks were given for partially corrected answers.

Problems:
Some answers did not provide a clear relationship of proximity. Indicating the CPU, either with text or drawing boxes as many students did would help to remove any confusion.

Several answers showed that the traits of the memory technologies were not understood although these were explained in lecture notes and were extensively discussed in lectures.

b)

This question was also quite popular. It only meant to check the understanding of the concepts of critical path delay, clock frequency, and the link between them. One mark was given for each part of the question. Accuracy in the numbers was not so important unless they were far off but units were important and marks were lost for such errors.

Problems:
Many answers reported unreasonable numbers. Frequencies in pHz(!) were reported. Other answers gave the frequency in seconds(!) which shows clear lack of understanding. No marks were given for such results even if arithmetic was right. Some flexibility for answers that reported MHz instead of GHz was shown depending on the quality of the solution.

c)

Another popular question but not so well answered. One mark was given for a reasonable definition of the timer and another mark was given for a logical description of its operation. Standard textbook question that did not require much thinking.

Problems:
Many seem to confuse the timer with the clock signal of the processor, for example, while it was emphasized that this is not the case.
Other answers stated that it is not the clock but did not describe what the timer is.
Several incorrect answers which had nothing to do with the operation of a timer.
d) About half of the class attempted this part. One mark was given for answering the DMA (and spelling it out) and another mark for giving the benefit (and overhead) of this peripheral.

Problems:
Some answers referred to caching, a topic not discussed in class and not included in the syllabus for this course as well.

Some answers mentioned registers. It is not clear how these answers relate to the question since the CPU will still be making the transfers.
Some answers mentioned only the name of the mechanism but did not describe the benefit/overhead of it.

e) Several attempted this part. Two full marks were given for the correct truth table, one mark for partially correct and no marks for largely erroneous answers. A similar example has been demonstrated in the last revision class.

Problems:
Most of the answers wrote down the entire truth table (16 cases) wasting time, the smart use of “don’t cares” (X) would have reduced the table to only 5 rows!
Some answers failed to consider that the INT signal is described as active low in the instructions and gave inverted answers. One mark was taken off for that if the rest of the thinking was correct.

f) Many attempted this answer. Although the initial intent was to answer the arithmetic operation of multiplication/division along with an appropriate arithmetic example, several answers referred to SER/DES and more specifically to the example of UART and drew its operation. This answer was also considered equally correct. One mark was given for describing the operation and another mark for a proper example.

Problems:
Several answered that shift registers do shifting! This is rather obvious and definitely not the sought answer.
Some gave an example with decimal numbers which was incorrect, since shift registers operate with binary numbers.
Other answers mentioned multiplication and provided as an example the full algorithm of full and long multiplication. This is exactly what we want to avoid with shift registers, so that this is not an appropriate example.
g) This part was also relatively popular. Most of the students have studied this in depth. One mark was given for giving the correct number and name of USB signals and some discussion and another mark for describing the principle of bit stuffing. If the exact number of bits was not given, marks were not taken off as long as the use of bit stuffing was correctly described.

Problems:
Nothing significant as no great misunderstandings or confusion about this answer was noticed. Some made random guesses about bit stuffing but these were faulty. Few confused the USB protocol with memory!

h) This question required some critical thinking and it was not just about recalling textbook material. Sadly few attempted it. The idea was to think inversely as compared to the example given to speed up MU0 by adding the register DIN. Some students got this exactly right, however, and gave nice explanations. I would like to thank them for that.
Section B

Answer one question from Q3 and Q4.

Q3 was by far the most popular question in this section with almost everyone answering it.

Q3

A popular question - almost everyone attempted it with mixed success. Overall, a decent attempt though – around 54% overall. However, there were a number of very poor answers. The question tested your understanding of the operation and design of finite state machines, and how to translate designs into Verilog, which is something you have ALL had experience of doing in the lab. Consequently, I would have felt it would have been a relatively straightforward question to answer. How wrong I was ...

a)

Part a) required you to identify the errors in a state transition diagram. There were six in total:

- Boolean conditions out of state_0 are not all satisfied
- No Boolean conditions for the transitions out of state_1
- Hung state_2 – how do you get out?
- state_2 and state_4 have the same state code
- Boolean conditions out of state_4 are not all satisfied
- state_6 invalid state code – it should be 3 bits

Problems:

- Only giving 3 errors – yes, there were 3 marks available, but that doesn't mean there are 3 errors!
- The state code for state_6 is NOT the same as state_1
- Reset comes from nowhere as a squiggly line into state_0 – what? This illustrates an asynchronous reset – we have covered this in lectures.
- The state codes are not in numerical increasing order – they don't have to be! When have I said this? In fact, example are given in lectures where the state codes are randomly assigned, which is perfectly fine. Generally however, the initial state should be given a 0 state vector, i.e. 000.
- No condition for z=1 for state_6 – it's not shown for simplicity. If a condition is not shown, then it implies you stay in the state for that condition – again covered in lectures.
- My favourite incorrect mistake was – conflicting conditions for leaving state_3 since DeMorgan's theorem states \[ a \cdot b = \overline{a} + \overline{b} \] – someone needs to refresh their understanding of DeMorgan's theorem!
b)

You are required to discuss the operation of the three functional blocks in the implementation of the FSM, i.e.

![Diagram of FSM blocks](image)

A mark was awarded for a suitable discussion for each, i.e.

- A combinatorial block of logic that determines the next state from the current state and any inputs
- A sequential block that assigns the value of next state to the current state at each rising edge of the clock
- A combinatorial logic block that determines any outputs from the current state.

I was actually quite generous with this question and awarded marks based on the production of a correct diagram (as shown above), without an accompanying discussion.

Problems:

- Not mentioning combinatorial or sequential
- Not being clear what output is produced and for what input(s) for each block
- Complete random discussions – for example, a functional block implementing reset – what?

c)

This proved an interesting question. You were simply required to translate the working state transition diagram into a state transition table (3 marks), and assign state codes to each state (1 mark). A suitable state transition table is given below. I didn’t drop marks if a default row wasn’t given.
<table>
<thead>
<tr>
<th>Current State</th>
<th>Inputs</th>
<th>Next State</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A B C</td>
<td>10p 20p</td>
<td>A' B' C'</td>
<td>ready  reject dispense</td>
</tr>
<tr>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>1 0 0</td>
</tr>
<tr>
<td>0 0 1</td>
<td>1 0 1</td>
<td>0 1 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>0 1 0</td>
<td>0 2 3</td>
<td>0 1 1</td>
<td>0 0 0</td>
</tr>
<tr>
<td>0 1 1</td>
<td>x x</td>
<td>0 0 0</td>
<td>0 0 1</td>
</tr>
<tr>
<td>1 0 0</td>
<td>x x</td>
<td>0 0 0</td>
<td>0 1 0</td>
</tr>
<tr>
<td>default</td>
<td>x x</td>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
</tbody>
</table>

Problems:

- Not listing transitions for conditions where 10p = 0 and 20p = 0 in the 0p_state, 10p_state, and 20p_state states.
- Showing reset in the table – when have we shown this?
- Giving a non-standard state transition table – random, made-up tables.
- Setting outputs to don’t cares– I stressed in lectures that outputs should always be set to a value, 0 or 1.
- Redrawing the state transition diagram simply to annotate the state codes – why waste time?
- Setting the outputs for the next state rather than the current state.

This involved taking the design from part c) and translating it into a Verilog implementation. The question actually tells you to implement this as three functional blocks (see part b), which implies three always blocks. The question does not require you to define the module, just to produce the three always blocks. Marks were awarded as follows:

- Next state logic – 4 marks
  - 1 mark for a sensitivity list with *, or all inputs listed
  - 1 mark for the use of blocking statements
  - 1 mark for a default – a case statement is the best implementation approach (here default should default to state 000)
  - 1 mark for being functionally correct when defining the next state value
- Current state assignment – 3 marks
  - 1 mark for the correct sensitivity list – posedge clk and posedge reset
  - ½ mark for a correct reset action
• ½ mark for a correct current state assignment
• 1 mark for the use of non-blocking assignments

• Output assignment – 3 marks
  • 1 mark for correct sensitivity list with *, or current state
  • 1 mark for blocking assignments
  • 1 mark for outputs being functionally correct

Problems:
• Not providing the answer as three blocks as instructed. There were quite a lot of answers where an attempt was made to merge the two or more blocks in a single always block. This is perfectly fine, with experience, however, you have NEVER been shown how to do this, so why do it? This often results in state changes and outputs being out of sync. Marks were often lost as a result!
• Using nested if else statements, invariably with dangling elses, rather than a case statement.
• What’s a “break” in a case statement?
• Having a synchronous reset action by not having a timing dependency on reset in the sensitivity list for the current state assignment.
• Having no idea what the difference is between blocking and non-blocking assignments and when it is appropriate to use them!
• Only setting outputs to 1, never setting them to 0. In a combinatorial block you should set ALL outputs to a value for ALL cases.
• General poor, incomprehensible, made-up Verilog code ...
Q4

Unfortunately, only a few students attempted this question, which was a bit disappointing as the first two parts were explained thoroughly in regular and revision lectures, the third was closely related to one practical exercise in the lab, and only the last part required some critical thinking based on the material taught in the module. Marked out of 20.

a)

A standard questions where several examples were given in the class. Most of the answers determined right the number of chips but the majority couldn't remember how a decoder works and provided the drawing of a MUX, which was not correct. Note a separate CS (chip select) signal is needed for each chip!

b)

The majority of the students who attempted this part got it fully or partially right, where the examples in the class helped a lot to understand the timing of MU0. Thank you! One part of the question was asking to provide the mnemonic of the instructions and then to fill out the table. This intended to help the students who could determine the instructions but not the timing to receive some marks. Some answers ignored this part and directly filled out the table. In principle, they should have not done so. Only one mark, however, was taken off form the overall question as long as the table was correctly filled in.

c)

Most of the students who tried this part got it right as it was based on the lab exercises. The schematics were mostly correctly drawn and the faulty connection was detected.

d)

Only a couple of students managed to solve this correctly. The idea was to recursively apply the given formulas to determine the Boolean expression for the carry out and then draw a circuit diagram. Adding up the delay of the gates on the longest path would yield the required answer. However, no path delay was determined correctly, the terms $G$ and $P$ add some delay to the drawn circuit diagram. This delay could have been readily determined as the logical expressions for each of these terms was given (involved only one gate) and the delay of these gates was given in the instructions.