2016 Paper E2.1: Digital Electronics II- Solutions

Q1 (a) This question tests students' understand the basic of FPGA and Verilog. They have already encountered the simple hex-to-7segment decoder in the lecture and the lab. This is an extension to that.

(i) This solution assumes that the tenth display is blank instead of showing a 0 when in ≤ 9:

```
module hex_to_BCD (dgt_1, dgt_0, in);
// this solution assumes tenth digits are blank or shows 1 if in>9
   output [6:0]   dgt_1, dgt_0;  // two display digits
   input [3:0]    in;            // binary inputs

   reg [6:0]      dgt_1, dgt_0;

   assign {dgt_1[6:3],dgt_1[0]} = 5'b11111; // all these segments are off
   always @(in)
      case (in)
      4'h0: {dgt_1[2:1], dgt_0}= 9'b111000000;
      4'h1: {dgt_1[2:1], dgt_0}= 9'b111111001;
      4'h2: {dgt_1[2:1], dgt_0}= 9'b110100100;
      4'h3: {dgt_1[2:1], dgt_0}= 9'b110110000;
      4'h4: {dgt_1[2:1], dgt_0}= 9'b110011001;
      4'h5: {dgt_1[2:1], dgt_0}= 9'b110010010;
      4'h6: {dgt_1[2:1], dgt_0}= 9'b110000010;
      4'h7: {dgt_1[2:1], dgt_0}= 9'b111111000;
      4'h8: {dgt_1[2:1], dgt_0}= 9'b110000000;
      4'h9: {dgt_1[2:1], dgt_0}= 9'b110010000;
      4'ha: {dgt_1[2:1], dgt_0}= 9'b001000000;
      4'hb: {dgt_1[2:1], dgt_0}= 9'b001111001;
      4'hc: {dgt_1[2:1], dgt_0}= 9'b000100100;
      4'hd: {dgt_1[2:1], dgt_0}= 9'b000110000;
      4'he: {dgt_1[2:1], dgt_0}= 9'b000011001;
      4'hf: {dgt_1[2:1], dgt_0}= 9'b000010010;
      endcase
endmodule
```

[6]

(ii) If one consider those outputs that are permanently '1' do not need logic, then there are only 9 outputs that are dependent on 4 inputs, we only need 9 LEs. However, I will accept answers between 9 and 14 with justifications.

[2]

(Although this question is an extension to what was in the laboratory session, some students found it difficult to handle the decoding for Digit_1. For those who could do it, they did very well.)

Q1 (b) This question tests students' understanding of the timing constraints in digital circuits.

(i)

Ignoring any hold time violation, the path from Q1 (FF1 Q output) to D2 governs the worst-case timing. This gives:

$$t_{clk} \geq 2 * t_{d(max)} + t_{cq(max)} + t_{su}$$

Therefore $t_{clk} \geq 2 * 1ns + 1ns + 1.3ns = 4.3ns$

Hence $f_{clk}(max) = 232.6MHz$.

[3]

(ii)

The shortest delay from Q2 (FF2 Q output) to D1 (FF1 D input) is:

$$t_{cq(min)} + t_{d(min)} = 1.0 \text{ ns}$$

which is shorter than the hold time requirement of FF1, i.e. 1.1ns. Therefore there is hold time violation at FF1 D input.

[3]

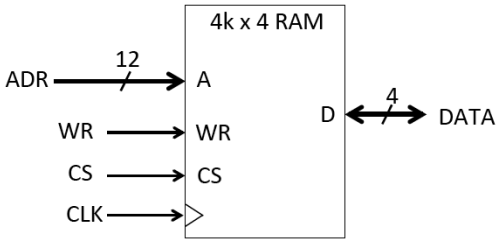
(iii)

To eliminate the hold time violation, we want to delay the arrival time at D1 by at least 0.1ns. This can be achieved by delaying to clock to FF2 by the same amount of 0.1ns. Therefore the minimum value required for t_{dly} is 0.1ns.

[2]

(I was glad to see that many students could answer this question. The feedback signal around the circuit made this question a bit hard as a Q1 problem, which is intended to be the "mastery" question, easier than Q2 and Q3. However, the question really tells me who understood timing constraints in digital circuits, and who did not.)

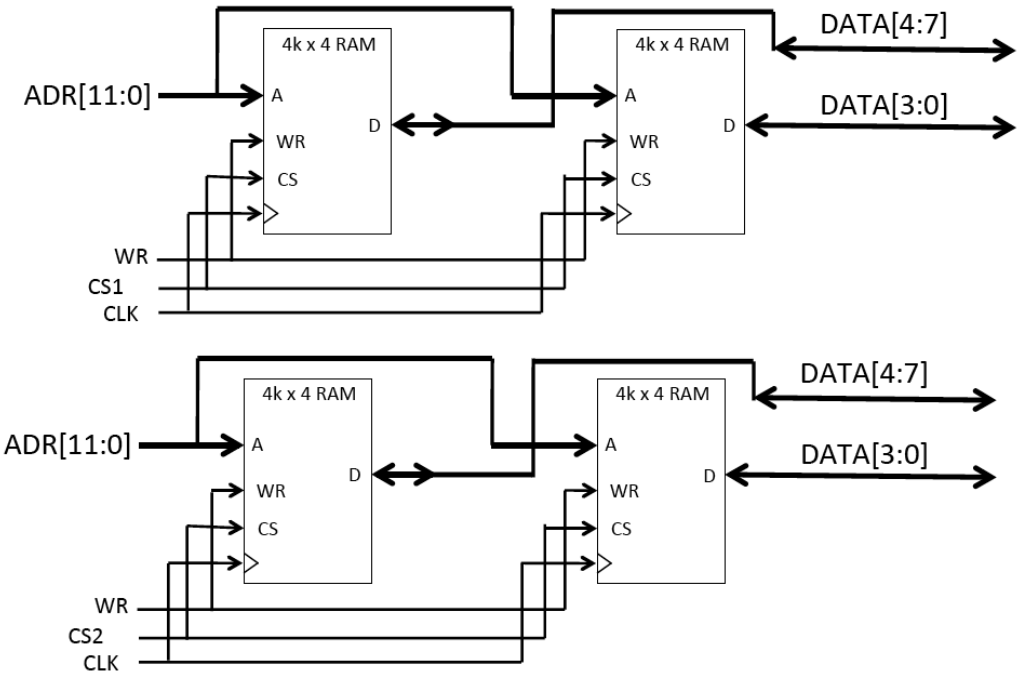
Q1 c) This question tests students' understanding of memory organisation and address decoding.



(i) Address: 12 bits, Data 4 bits

[2]

(ii) a)



[4]

b) CS1 address range: 16'h8000 to 16'h8fff.

CS2 address range: 16'h9000 to 16'h9fff.

Therefore:

CS1 = A15 & !A14 & !A13 & !A12

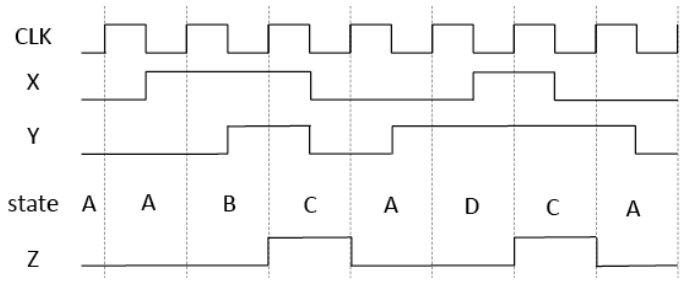
CS2 = A15 & !A14 & !A13 & A12

[2]

(Many students found this question difficult – which is a surprise to me. I think I need to provide more tutorial problems on how memory could be organised in a digital system. This question is unlike previous questions in the past paper. The "twist" in the question resulted in some students not able to answer the question.)

Q1 (d) This tests students' ability to work with FSM and to specify a simple FSM in Verilog.

(i)



[4]

(ii) Note that only the code for S_A in the case statement is required.

[4]

```
module fsm1 (Z, X, Y, CLK);

output   Z;
input    X, Y, CLK;

// define the states - one-hot encoding
parameter S_A = 4'b0001;
parameter S_B = 4'b0010;
parameter S_C = 4'b0100;
parameter S_D = 4'b1000;

reg [3:0]   state;
reg         Z;

wire        CLK, X, Y;

initial     state = 0;
```

```
// specify state machine
always @ (posedge CLK) begin
   Z <= 1'b0;      // default Z value
   case (state)
      S_A:  if ((X==1'b1) && (Y==1'b0))
               state <= S_B // XY = 10 -> goto B
            else if ((X==1'b0) && (Y==1'b1))
               state <= S_D // XY = 01 -> goto D
            else if ((X==1'b1) && (Y==1'b1)) begin
               state <= S_C; // XY = 11 -> goto C
               Z <= 1'b1;
               end
            else  state <= S_A; // stay in A

      S_B:  if ((X==1'b0) && (Y==1'b0))
               state <= S_A; // XY = 00 -> goto A
            else if ((X==1'b1) && (Y==1'b1)) begin
               state <= S_C; // XY = 11 -> goto C
               Z <= 1'b1;
               end;
            else  state <= S_B; // stay in B

      S_C:  state <= S_A; // always goto A

      S_D:  if ((X==1'b0) && (Y==1'b0))
               state <= S_A; // XY = 00 -> goto A
            else if ((X==1'b1) && (Y==1'b1)) begin
               state <= S_C; // XY = 11 -> goto C
               Z <= 1'b1;
               end
            else  state <= S_D; // stay in D
      default: state <= S_A;
   endcase
endmodule
```

(Students did well on this question. Some students did not read the question carefully and waste time in provide the full Verilog code, when only the segment for state A is required. Nevertheless, most could answer this part very well.)

Q1 (e) This tests students' understanding of DAC principles, terms and function.

(i) Basic bookwork. The resistor network divides 5V into 1024 voltage values. IN[9:0] is decoder into a one-hot code, which turns ON one of the 1024 switches to route the appropriate voltage from the resistor-divider to the input of the unity gain amplifier. The decoder circuit takes the digital input and set the corresponding output BIT to 1 and the rest to 0 (one-hot code).

[2]

(ii) Vout = 5 x (IN[9:0]+1) / 1025 (volts)

[2]

(iii) Range = 4.89mV to 4.995V, resolution = 4.89mV

[2]

(iv) IN = 10'b0100011010 = 10'h11a. Therefore in decimal IN = 1*256 + 1*16 + 10 = 282.

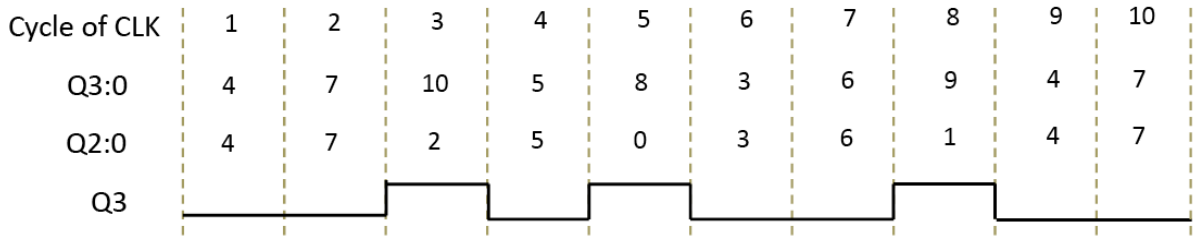
Therefore SW_{282} is turned ON, and Vout = 5*(282+1)/1025 = 1.38V.

[2]

(Those who can answer this question got perfect marks. It is perhaps the easiest question in the whole paper.)

Q2 This question tests students' ability to walk through the working of a reasonably complex digital circuit, then derive its function. It also tests students' understanding of how circuits could be implemented on FPGAs, and the resources that would be used. Finally it tests students' ability to estimate how fast the circuit might work.

(a) At the end of each clock cycle, Q3:0 = (Q3:0 mod 8) + 3. Hence we get:



[10]

(b) Q3:0 (initial) = 0, sequence of Q3:0 is 0, 3, 6, 9, 4, …

Q3:0 (initial) = 1, sequence of Q3:0 is 1, 4, …

Q3:0 (initial) = 2, sequence of Q3:0 is 2, 5, 8, 3, 3, 6, 9, 4, …

Q3:0 (initial) = 3 and 6 are both a subset of Q3:0 = 0

Q3:0 (initial) = 5 is a subset of Q3:0 = 2

Therefore all cases ended up with the same repetitive cycle as Q3:0 = 4.

[5]

(c) Looking at (i), it is clear that Q3:0 repeats itself after 8 cycles and produces 3 pulses on Q3. Therefore average frequency of Q3 is (3/8) x 50MHz = 18.75MHz.

X3 has a frequency one sixteenth of Q3, therefore its average frequency is 18.75/16 = 1.172MHz.

[5]

(d)

M1: 3 LEs (each LE can do one bit adder)

M2: Since each LE also has a D-FF, M2 is part of M1 LEs

M3: Each counter bit requires one LE, therefore 4 LEs

Total LE usage is 7.

[5]

(e)

Assumption is that CTR4 is not he critical component to determine operating frequency. Not knowing exactly what's inside CTR4, that's the only assumption one can make.

The adder has 3 bits to propagate. Therefore the worst-case delay is: $t_{cq} + 3*t_{lut} + t_{su} = 0.1 + 3*0.25 + 0.08 = 0.93$. Therefore this circuit can work up to 1075MHz. Hold time does not matter.

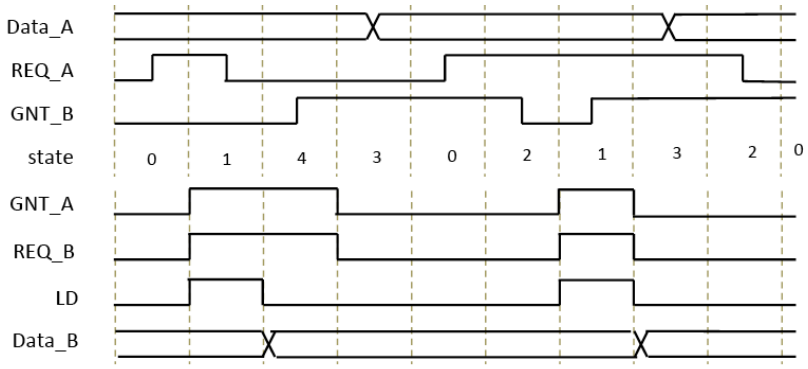
Further, we also use CO from the adder to drive M2, therefore students may assume $4 * t_{lut}$ delay, which is also acceptable. In which case the total worst-case delay is 1.18ns and max freq = 847.5Hz. Both answers are acceptable.

[5]

(Most student could do part a) well. Some students managed a perfect score on the other parts of the question. Therefore this question is definitely “do-able”. The other parts of the questions achieved the goal of separate those with depth of understanding from those who did not. Overall, students did well on this question.)

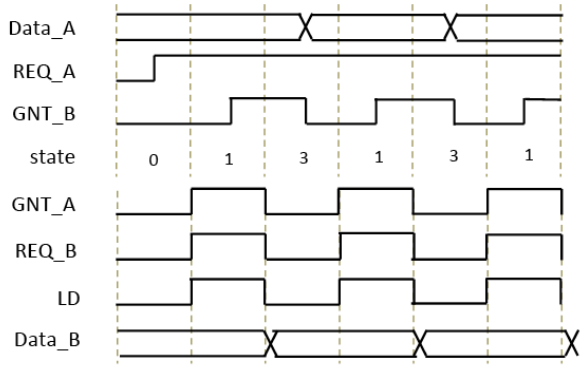
Q3 This question tests students' understanding of interfacing between two systems and how to analyse and design a more complex FSM.

(a)



[15]

(b) Assumption is that REQ_A remains high and GNT_B is asserted as soon as REQ_B goes high. Under this circumstance, new data is transferred every two clock cycles.



[15]

(Many students found this question difficult. In spite of that, many did part a) perfectly. A small number of students answered this question perfectly. This is definitely the hardest question in the whole paper.)

OVERALL COMMENT FOR THE PAPER

The result for this paper is near perfect. Question 1 has an average of 29.9/40, which is nearly exactly as the goal of 30/40. Q2 is easier and Q1 is harder. However, the overall average of the paper is 62%, which is within the intended bound of 55% to 65%. Finally, only 6 students failed this paper. They clearly have not met the minimum threshold expected in this subject.