2018 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% each

Time allowed: 2 hours

SOLUTIONS

(For Examiners Only)

1. (a) This question examines student's understanding of digit circuit timing including setup and hold times, and the idea that inserting pipeline registers between logic potentially increases the system clock frequency.

(i) $$\max(t_{c-q} + t_{lut} \times 2 + t_{setup}) \leq \min(t_{clk} + t_{buffer})$$

[4]

(ii) $2 + 2x5 + 2 \leq tclk + 2$, therefore $tclk \geq 12$ ns. Fmax = 83.3MHz

[2]

(Most students got part (i) and (ii) correct. Some made the mistake of putting t_{buffer} in the wrong side of the inequity.)

(iii) Clock frequency can be increased by inserting a flip-flop between LUT_1 and LUT_2. Assuming that the inserted FF is also clocked by the same clock signal as FF2, and have the same timing constraints as FF1 and FF2, then the setup time constraint becomes:

$2 + 5 + 2 \leq tclk + 2$, and $tclk \geq 7$ ns, Fmax is now = 143MHz.

[2]

(This caught out many students who did not realise that inserting a flip-flop, i.e. adding pipeline registers, will increase clock frequency at the expense of adding an extra clock cycle of latency. A reasonable number of students got this one right, but the majority did not.)

(b) This question tests students understanding of shift register and LFSR etc. It also tests the ability to use Verilog to specify a shift register circuit.

(i)

Q4:Q3:Q2:Q1	Q4^Q1
0 1 0 0	0
1 0 0 0	1
0 0 0 1	1
0 0 1 1	1
0 1 1 1	1
1 1 1 1	0
1 1 1 0	1
1 1 0 1	0
1 0 1 0	1
1 0 1 1	0
0 1 1 0	0

[2]

(This is an easy question that most students got right.)

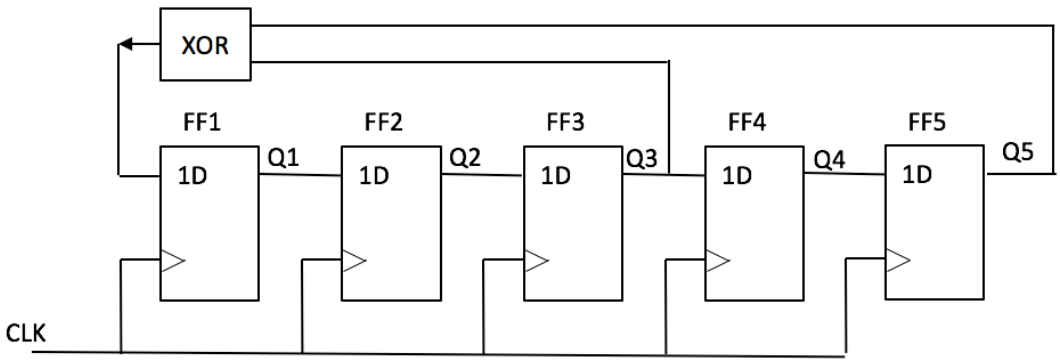
(ii)

```
module sr4 (
   clk,
   q);

   input clk;
   output [4:1]  q;

   reg [4:1] sreg;

   initial sreg = 4'b0100;

   always @ (posedge clk)
      begin
         sreg[4:2] <= sreg[3:1];
         sreg[1] <= sreg[4] ^ sreg[1];
      end

   assign q = sreg;

endmodule
```

[4]

(Again, most student were able to do this correctly.)

(iii)



[2]

(Most students also got this correct. Note that I deliberately used Q1 to Q5, not Q0 to Q4 because this matches the power of the primitive polynomial.)

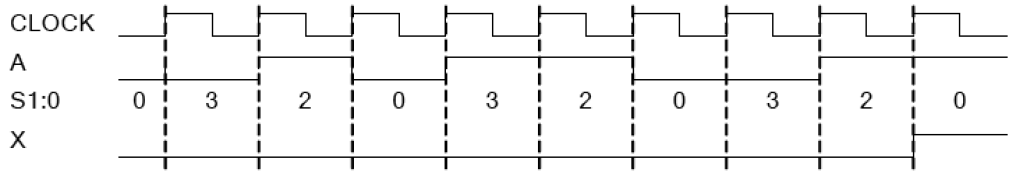
(c) This tests student's understanding of FSM.

(i)

	D1:0/X	A=0	A=1
S1:0	00	11/0	01/1
	01	10/0	00/0
	10	00/0	00/0
	11	10/0	10/0

[2]

(ii)



[2]

(iii)

```
module FSM (CLK, A, X);
   input CLK, A;
   output X;

   // declare and define states
   reg [3:0]   state;
   parameter   S0 = 4'b0001, S1 = 4'b0010, S2 = 4'b0100, S3 = 4'b1000;

   initial state = S0;

   // state transitions
   always @ (posedge CLK)
      case (state)
         S0:   if (A==1'b1) state <= S1;
               else state <= S3;
         S1:   if (A==1'b1) state <= S0;
               else state <= S2;
         S2:   state <= S0;
         S3:   state <= S2;
         default: state <= S0;
      endcase

   // FSM output
   always @ (*)
      if (state==S0) X = A;
      else X = 1'b0;

endmodule
```

[4]

(Most students found this question straight forward. Part iii has many different solutions. This one is just an example solution.)

(d) This question tests student's understanding of memory map and address decoding circuits.

(i) The address ranges for the four spaces are:

ROM_CS:	18'h00000	to	18'h07FFF
NVRAM_CS:	18'h08000	to	18'h0BFFF
RAM_CS:	18'h10000	to	18'h1FFFF
IO_CS:	18'h3FA0	to	18'h3FEF

[4]

(Surprisingly, quite a number of students found this problem difficult. In particularly, many got IO_CS wrong!)

(ii)

```
module decoder (
    addr, rom_cs, nvram_cs, ram_cs, io_cs);

    input [17:0] addr;
    output  rom_cs, nvram_cs, ram_cs, io_cs;

    assign rom_cs = ~addr[17]&~addr[16]&~addr[15];
    assign nvram_cs = ~addr[17]&~addr[16]&addr[15]&addr[14];
    assign ram_cs = ~addr[17]&addr[16];
    assign io_cs = addr[17]&addr[16]&addr[15]&addr[14]&addr[13]
                   &addr[12]&addr[11]&addr[9]&~addr[8];

endmodule
```

[4]

(Students who got (i) right tends to also get (ii) correct.)

(e) This question examines student's understanding of propagation delay in adder circuit and how carry-skip scheme could help in reducing propagation delay.

(i) P0 to C2A = 2 + 2 + 2 = 6ns
C-1 to C2A = 2 + 2 + 2 = 6ns

$P0 \rightarrow C2B = \max(2 + 2 + 2 + 2 , 2 + 1 + 3) = 8$

$C_{-1} \rightarrow$ = C2B =2

The following explanation is not expected, but it is include here for educational purpose in future years.

Note that although there is an apparent path $C_{-1} \rightarrow C0 \rightarrow C1 \rightarrow C2A \rightarrow C2B = 8$.

This in fact never actually arises - it is never true that a change in C-1 will cause this chain of cause and effect because it would require all the full-adders to be propagating the carry (to enable $C_{-1} \rightarrow C0 \rightarrow C1 \rightarrow C2A$).

Also SEL=0 (to enable the final step C2A → C2B). The entire point of the circuit (known as carry-skip adder) is that we prevent this from happening. Normally when analysing a circuit for the worst-case propagation delay, we assume that any path that is present in the circuit will be enabled for some set of input conditions; the carry-skip circuit is one of the rare cases when this assumption is untrue.

[4]

(ii)
This circuit suggests that C2B may have a longer worst case delay. However, if multiple states of these car cascade together, the C-1 to C2A paths is the worst case delay, which get added together (carry propagate). With carry-skip circuit here, the C-1 to C2B (carry out) is reduced from 6ns to 2ns (skip circuit). Therefore two case cascade with reduce C-1 to Cout delay from 12 ns to 4 ns.

[4]

(Most students found this question difficult. I guess this is because they have not encountered this type of circuits before. Quite a number of students got part (i) but not part (ii).)

Overall comments for Q1:

Most students did well in Q1. The average score for Q1 was 28/40, with a standard deviation of 4.3 marks.

2. This questions tests student's understand of ADC, voltage range and idea of accuracy as expressed in LSB.

(a)

ABCD	Vin
0000	Vin ≤ 0
0001	0 ≤ Vin < 1
0011	1 ≤ Vin < 2
0111	2 ≤ Vin < 3
1111	Vin > 3

[10]

(b)

```
module encoder (
    A, B, C, D, X);

    input A, B, C, D;
    output [2:0] X;

    reg [2:0] X;

    always @*
       case {A,B,C,D}
          4'b0000: X <= 3'd0;
          4'b0001: X <= 3'd1;
          4'b0011: X <= 3'd2;
          4'b0111: X <= 3'd3;
          4'b1111: X <= 3'd4;
       default
          x <= 3'd0;
       endcase
endmodule
```

[10]

(c) Worst-case scenario is when there is maximum mismatch in resistor values. Consider two scenarios:

1) the bottom R is -20% and the other two are +20%, the threshold for 0000 to 0001 transition will change from 1V to 3*0.8/3.2 = 0.75V instead of 1V. Therefore error is 0.25LSB.

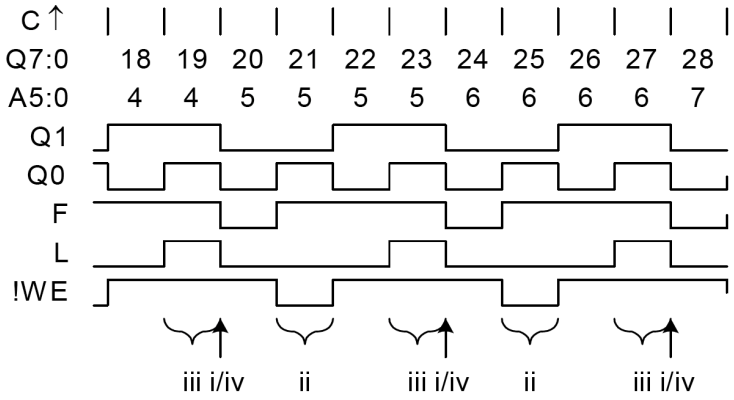
2) the bottom R is +20% and the other two are -20%, the threshold for 0000 to 0001 transition will change from 1V to 3*1.2/2.8 = 1.286 instead of 1V. Therefore error is 0.29LSB.

[10]

(Most students got part a) and b) more or less correct (with some minor errors). Very few students managed to do c). The average marks for Q1 was 19.8 out of 30, and the standard deviation was 5.5.)

3. This question tests student's ability to understand the operation of a fairly complex digital circuit with consisting of different types of components that they have encountered during the course and the Lab experiment, and able to work out its operating in term of the timing diagram.

(a)



[15]

(b)

X7:0 is stored on the rising edge marked "i" since L is high. It is then stored in address location 5 when !WE goes low at "ii". Since N5:0 = 0, the RAM address is still equal to 5 at "iii" when the value is read out of RAM and stored in the output register at "iv".

[5]

(c)

(i) When N5:0=0 as in the diagram, X7:0 will change on the rising edge of Q1at the start of cycle 18 and be output at the start of cycle 24 as described above. This gives a delay of 6 clock cycles.

(ii) When N5:0 = 63 = –1 (modulo 64), the value of A5:0 will be reduced by 1 when L goes high. This means that in the above diagram, the value of X7:0 that was written in cycle 21 will not be retrieved until cycle 25. This will add another 4 clock cycles onto the delay with will now equal 10 clock cycles.

(iii) We can deduce a general formula from the previous two answers:

$$\text{delay} = 6 + 4(64 - n) \mod 64$$

This correctly gives a minimum delay of 6 when n=0 and increments by 4 cycles if n is decreased by 1. It gives a delay of 258 clock cycles when n = 1.

[10]

(A significant number of students got part a) of the question at least partially correct. Very few managed to do b) and c). Average for Q3 was 17.2 out of 30. S.D. was 5.6)

Overall comment for this paper:

The class average was 64.9% - very close to the target of 65%, and the S.D. was 11.7%, which is also reasonable. Just over 1/3 of the class got A and B grades, 20% got C, 7% got D, and only 2 students failed.