IMPERIAL COLLEGE LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
EXAMINATIONS 2024

CIRCUITS AND SYSTEMS

Tuesday 14 May 14:30

Time allowed: 2 hours

SOLUTIONS

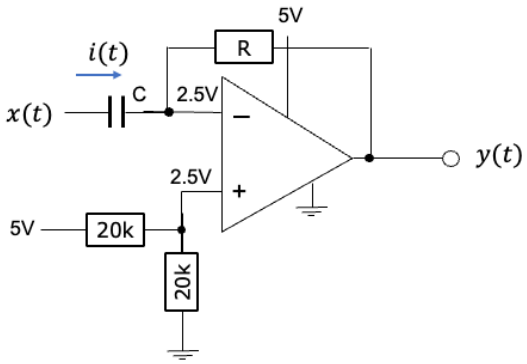
There are THREE questions on the paper.

Answer ALL questions.

1. (a) This question tests student's ability to apply an op-amp to implement a differentiator circuit and able to analyse the how the circuit works.

(i) The two resistors on the positive input of the op-amp provides an offset of 2.5V to the output. Current into the capacitor $i(t) = C\frac{d(x(t)-2.5)}{dt}$. Therefore,

$$y(t) = -RC\frac{dx(t)}{dt} + 2.5$$



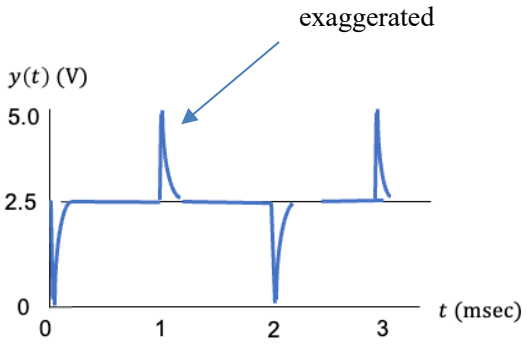
This is a differentiator circuit.

Also acceptable solution: $Y(s) = sRC\big(2.5 - X(s)\big) + 2.5$

[4]

(ii) Since R = 1k and C = 10nF, the time constant is RC = 10^{-5}, or 0.01ms. This is significantly shorter than the 1msec of the high and low period. Therefore, the behaviour of this circuit IS NOT a highpass filter, but edge detection. Further, the voltage across C cannot change instantaneously. Therefore, on rising edge of input, the -ve input of op-amp is forced to go +2V, thus forcing the output to go to 0V (-ve infinite gain). The output then decay to 2.5V with a time constant of 0.01ms. On the fall edge, the same happens except in the opposite direction.

Hence the output is:



[6]

COMMENTS:
Many students got this wrong because they treat the circuit as a filter (some as lowpass). However, due to the input signal being a square wave, the op-amp will not behave as a linear amplifier. Nevertheless, a small proportion of students got this perfectly!

(b) This question tests student's understanding of FSM, one-hot encoding and Mealey FSM that allows outputs to change mid-clock period.

(i) In one-hot encoding, each state is identified by one bit in the state code.

Therefore, an N-bit state machine would require N flip flops. In binary state encoding, the binary number is used to represent the state code. Therefore, an N-bit state machine would only need $\lceil log_2 N \rceil$, flip flops.

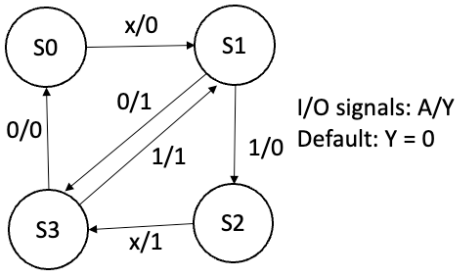
One-hot encoding is particularly suited to FPGA architecture because:

1) One-hot encoding requires fewer combinational logic because state decoding is not required. FPGA LUTs can generally only implement simple Boolean equations with few variables. This fine-grain architecture matches the simpler logic well. Also it minimize glitches – low power and more reliable.

2) FPGAs LEs contains D-FF which makes the architecture FF rich. Hence having to use more FFs in one-hot encoded state machine generally does not consume more hardware resources since the FF comes free.

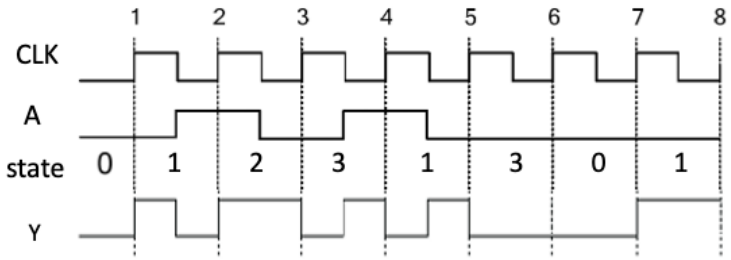
[3]

(ii)



[3]

(iii)

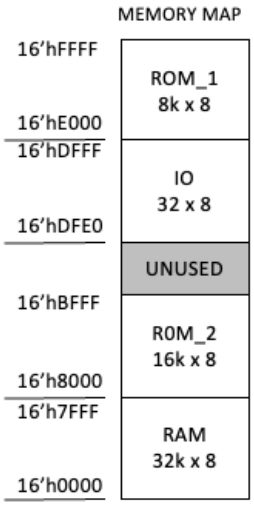


[4]

COMMENTS:
Full marks for those who make one of the justifications for using one-hot state encoding. Most students got the state diagram correctly, but many did not get Y waveform. This is a Mealey machine, therefore the output can change with input A, not just at rising edge of clock.

(c) This question tests student's understanding of memory address map, capacity of memory, address decoding and SystemVerilog specification of combinatorial circuits.

(i) The address map is:



[4]

(ii)

```
module addr_decoder (
    input  logic    [15:0]  A,       // address input
    output logic            ROM_1,      // ROM 1 select
    output logic            ROM_2,      // ROM 2 select
    output logic            RAM,        // RAM select
    output logic            IO          // IO select
)
    assign ROM_1 = A[15] & A[14] & A[13];
    assign ROM_2 = A[15] & !A[14];
    assign RAM = !A[15];
    assign IO = A[15]&A[14]&!A[13]&A[12]&A[11]&A[10]&A[9]&A[8]&A[7]&A[6]&A[5];
endmodule
```

[6]

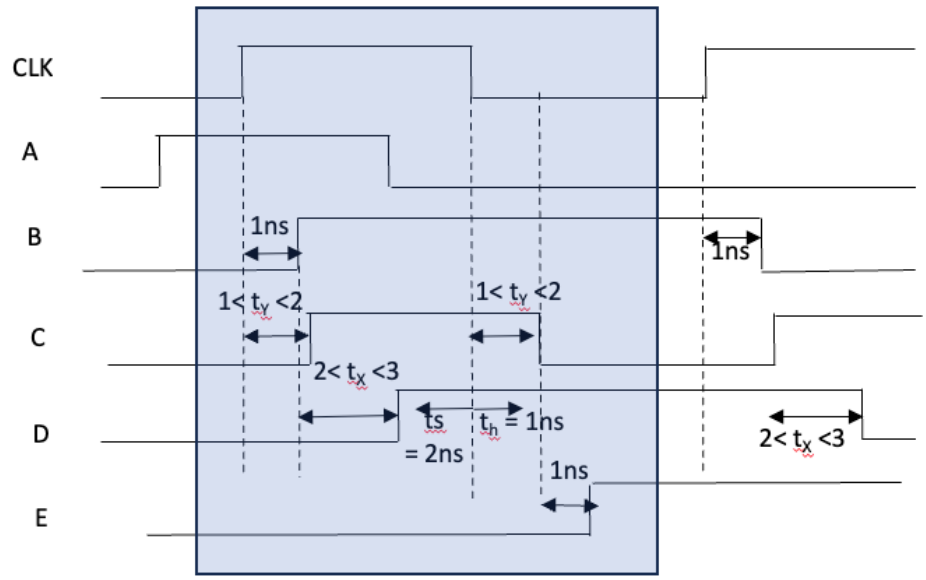
COMMENTS:

This is an easy question. A few student took a long time to work things out – because they are unfamiliar with hexadecimal and that 1k address is 16'h0 to 16'h3FF (10-bit).

(d) This question tests student's ability to work out timing constraints in a sequential circuit with delay in the clock path.

Note that the right FF is clock on the FALLING edge of the clock – hence clock period dictating timing is ½ T.

(i)



[4]

(ii)

Setup inequality:

$$t_p + t_{X\_max} + t_s < t_{Y\_min} + ½\ T$$

$$\Rightarrow\ 1 + 3 + 2 < 1 + ½\ T$$

T > 10ns

[4]

(iii) Hence, T >10ns, fmax < 100MHz.

[2]

COMMENTS:

Some students did not realised that the two FFs are clocked on opposite phase of the clock signal. However, many got the answer perfectly.

(e) This question tests student's understanding of R-2R ladder network of a DAC.

(i) This part of the question is purely bookwork.

Resolution – the voltage step equivalent to one least significant bit change of the digital number. Assuming that the digital number is N-bits, then this is the same as $\frac{full-scale\ voltage}{2^N-1}$.

Monotonic DAC – One that always goes up as the input number increases.

Settling time – Time taken to reach final value as input changes.

[3]

(ii) The resolution of a 10-bit DAC with full-scale voltage of 3.3V is 3.3V/1024 = 3.22mV. (3.3/1023 is also acceptable.)

[2]

(iii) $I_0 = 0.5$mA. The digital value is 10. Therefore V_{out} = -10 x 0.5m x 1k = -5V.

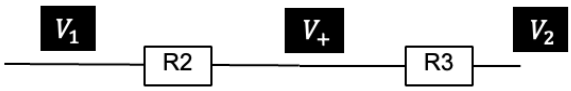
[5]

COMMENTS:

Some students did not know what monotonicity in a DAC means. However, most students got this question perfectly. Some missed out the negative sign for V_{out}.

2. This question tests student's understanding of using operational amplifier in a positive feedback loop to produces an oscillating function generator, and a pulse-width modulator circuit.

a) We can redraw the R2, R3 network as:



IC2 is a Schmidt trigger comparator circuit with threshold voltage at V_1 such that V_+ is at the reference voltage of 2.5V. Simple KVL give us this equation:

$$V_1 = V_+ \frac{R2 + R3}{R2} - V_2 \frac{R2}{R3}$$

V_+ = 2.5V, R2 = 100k and R3 = 200k.

When $V_2 = 0V,\ V_1 = 3.75V$.
When $V_2 = 5V,\ V_1 = 1.25V$.

[6]

b) For the integrator IC1, current flowing into the input is

$I_{in} = \frac{V_2 - 2.5}{R1+RV1} = \pm\frac{2.5}{R1+RV1}$ for V_2 = 5V or 0V respectively.

Since this is an integrator, the output voltage gradient: $\Delta V_1/\Delta T = -I_{in}$ /C1. Therefore

$$\Delta V_1/\Delta T = \mp\frac{2.5}{(R1 + RV1)C1}$$

Since V_1 integrates from 1.25V to 3.75V over half a period, $f_{max} = 200Hz$,

$\Delta V_1 = 2.5$ and $\Delta T = 0.5/f$.

At maximum frequency, $\Delta T = \frac{0.5}{200} = \frac{1}{400} sec$. Therefore, substitude into the equation above:

$$\frac{\Delta V_1}{\Delta T} = 2.5 \times 400 = \mp\frac{2.5}{R1C1}$$

Therefore **R1 = 250kΩ.**

To get $f_{min} = 166.7$Hz, $\Delta T = 1/333.4$.

$$\frac{\Delta V_1}{\Delta T} = 2.5 \times 333.4 = \mp\frac{2.5}{(R1 + RV1)C1}$$

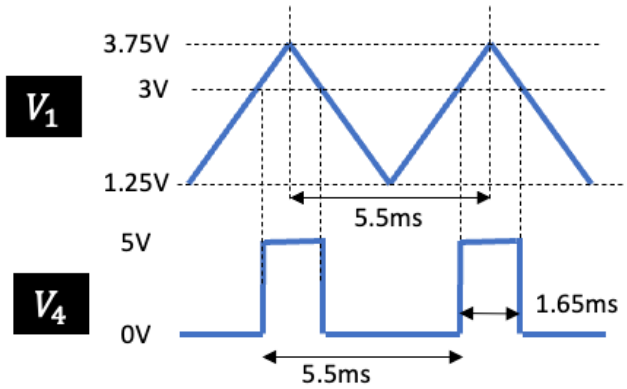
Therefore R1+RV1 = 300kΩ, **RV1 = 50kΩ.**

[12]

c) RV1 at mid-position means that it is 10kΩ. $f = 181.8$ Hz. The period is therefore 5.5ms.

The duty cycle of the PWM signal V_4 is independent of the frequency. Since V_3 is 3V, V_4 is high only when V_2 is above this value.

Since the gradient is constant, the duty cycle is simply $duty_{cycle} = \frac{3.75-3}{3.75-1.25} = 30\%$. Hence V_4 is high for 5.5 x 0.3 ms = 1.65ms.



[7]

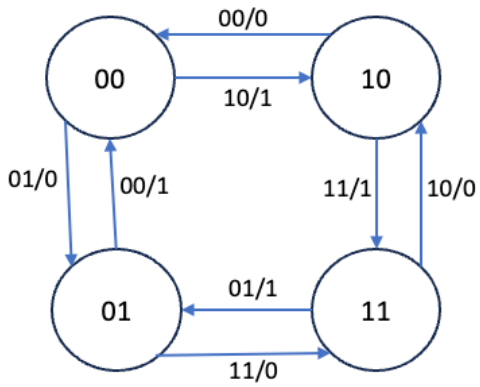
COMMENTS:

Most students got part a) perfectly. Part b) was a challenge to many. Instead of understanding what the entire circuit does and how the triangular wave generator works, they just treated the integrator as a low pass filter and the output frequency is the RC break frequency, which is totally wrong.

In part d) many did not work out the duty cycle and hence the time at which V4 goes high and low.

3. This question tests student's ability to design a finite state machine to solve a problem, and to build a small digital system to achieve some goal with standard components.

a)



Input: AB
Output: D
Default: D=0 (clockwise)

[10]

b)

```
module dir_fsm (
    input  logic    clk,    // clock
    input  logic    A,      // sensor A input
    input  logic    B,      // senosr B input
    output logic    D       // direction 0 - CW, 1 - CCW
);

// define states
    typedef enum {S00, S10, S11, S01} my_states;
    my_state current_state, next_state;

// state registers
always_ff @(posedge clk)
    current_state <= next_state;

logic [1:0] AB;     // define 2-bit input
assign AB = A|B;    // concatenate inputs
```

```
// next state logic
always_comb
    case (current_state)
        S00: if (AB==2'b10) next_state = S10;
             elseif (AB==2'b01) next_state = S01;
        S10: if (AB==2'b11) next_state = S11;
             elseif (AB==2'b00) next_state = S00;
        S11: if (AB==2'b01) next_state = S01;
             elseif (AB==2'b10) next_state = S10;
        S01: if (AB==2'b00) next_state = S00;
             elseif (AB==2'b11) next_state = S11;
             default: next_state = current_state;
    endcase

// output logic
always_comb
    case (current_state)
        S00: if (AB==2'b10) D = 1'b1;
             elseif (AB==2'b01) D = 1'b0;
        S10: if (AB==2'b11) D = 1'b1;
             elseif (AB==2'b00) D = 1'b0;
        S11: if (AB==2'b01) D = 1'b1;
             elseif (AB==2'b10) D = 1'b0;
        S01: if (AB==2'b00) D = 1'b1;
             elseif (AB==2'b11) D = 1'b0;
             default: D = 1'b0;
    endcase
endmodule
```

[15]

COMMENTS:
Quite a few students did not use a clock signal to drive the FSM, and instead was very unsure exactly how to use A and B. The best solutions (better than even the one above) is to use two separate outputs CW and CCW, instead of one shown here. In which case, the output could be CW, CCW, or unsure!