s 2015 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 the schematic diagram of a 16-bit binary counter implemented on an Altera Cyclone III FPGA. It consists of a 16-bit adder and sixteen D-flipflops. EN is the enable input: if EN = 1, the counter output CTR[15..0] is incremented by 1 on the rising edge of CLK; otherwise, the counter is not incremented.

It is known that each Logic Element (LE) contains a 4-input lookup table (LUT) and a D-flipflop. The LUT has a worst-case delay of 250 ps and the D flipflop has a clock-to-output delay of 100ps, and a setup and hold time of 80 ps and 45 ps respectively.

(i) Estimate and justify the number of Logic Elements (LEs) required to implement this binary count.

[2]

(ii) Calculate the maximum clock frequency at which this counter will operate.

[3]

(iii) Specify the counter in Verilog HDL that can be synthesized.

[3]

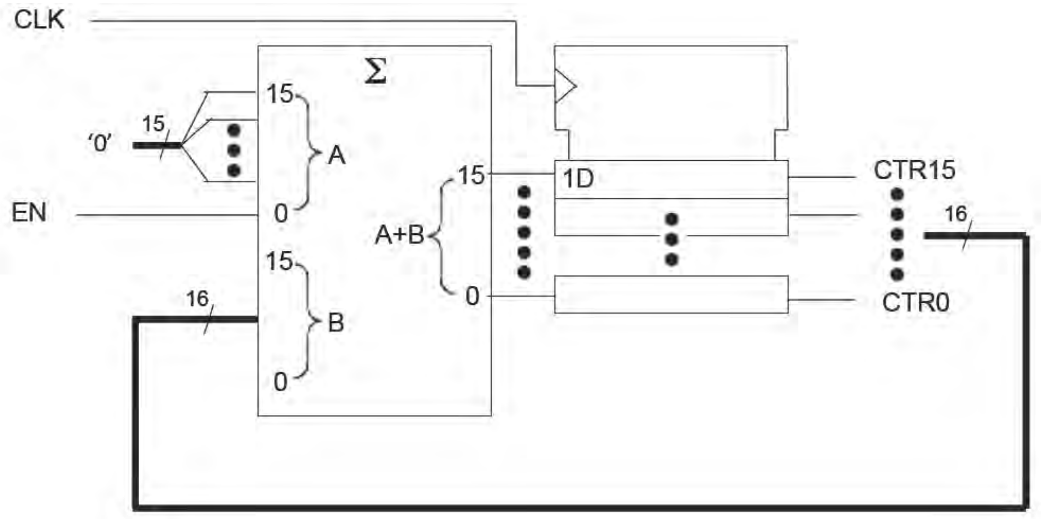


Figure 1.1

(b) (i) Explain the meaning of the following specifications for a digital-to-analogue converter (DAC): resolution, linearity error, monotonicity and settling time. Give one example where the monotonic behaviour of a DAC is important.

[4]

(ii) *Figure 1.2* shows a circuit for a R-2R 4-bit DAC with input IN[3:0] = 4'b0110. The resistor network is supplied via a 8mA ideal current source. The digital inputs IN[3:0] control the four 2-way electronic switches. The output V_{out} is driven by an ideal operational amplifier.

Derive the values for the current I_0, and voltages V_0, V_2 and V_{out}.

[4]

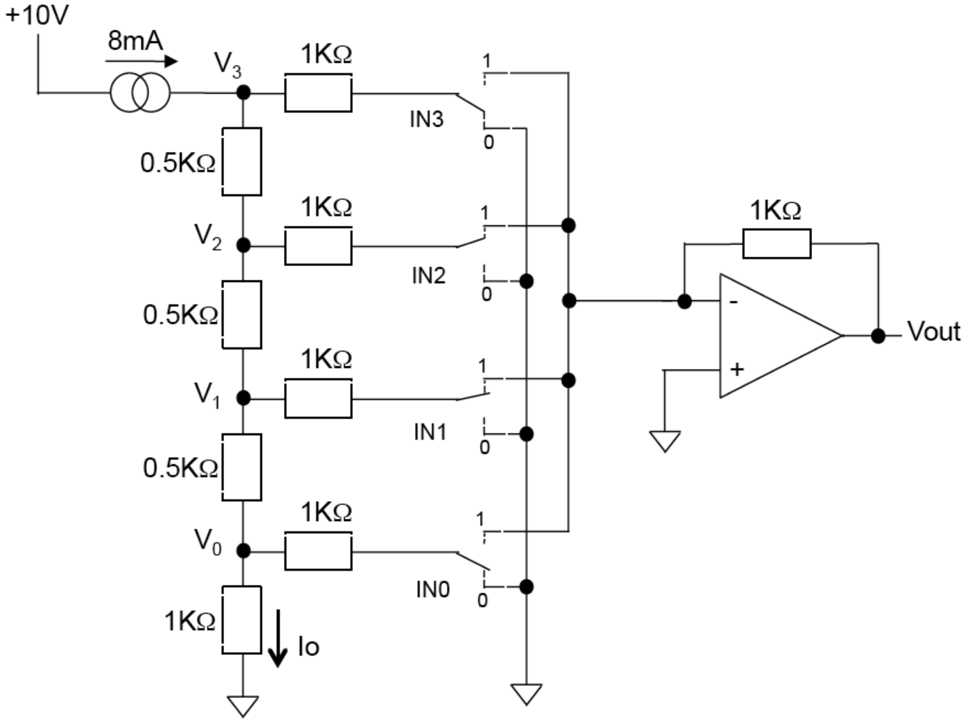


Figure 1.2

(c) *Figure 1.3* shows the Verilog specification of a finite state machine FSM with four states that uses one-hot encoding.

(i) What is one-hot encoding? Briefly explain why one-hot encoding is particularly suitable for FPGA implementation.

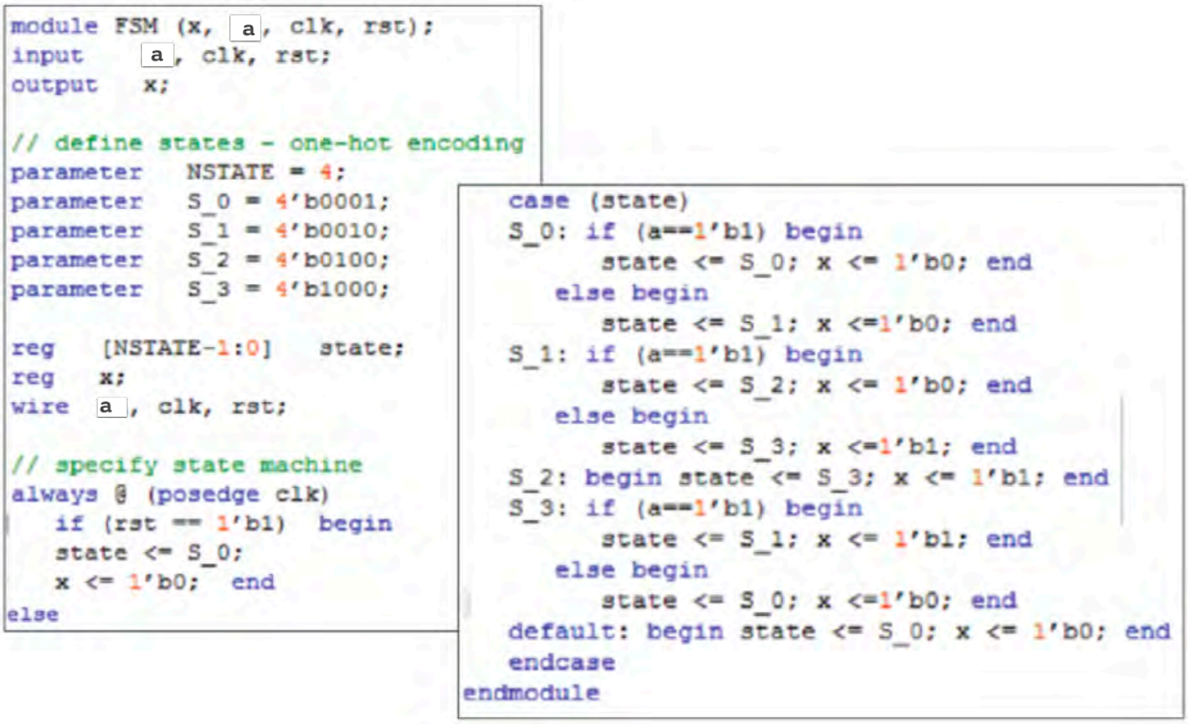
[3]

(ii) Construct the state transition diagram for this FSM.

[3]

(iii) Hence or otherwise, complete the timing diagram shown in Figure 1.4.

[2]



```
module FSM (x, a, clk, rst);
input     a, clk, rst;
output    x;

// define states - one-hot encoding
parameter   NSTATE = 4;
parameter   S_0 = 4'b0001;
parameter   S_1 = 4'b0010;
parameter   S_2 = 4'b0100;
parameter   S_3 = 4'b1000;

reg   [NSTATE-1:0]   state;
reg   x;
wire  a, clk, rst;

// specify state machine
always @ (posedge clk)
   if (rst == 1'b1)  begin
   state <= S_0;
   x <= 1'b0;  end
else
```

```
   case (state)
   S_0: if (a==1'b1) begin
            state <= S_0; x <= 1'b0; end
         else begin
            state <= S_1; x <=1'b0; end
   S_1: if (a==1'b1) begin
            state <= S_2; x <= 1'b0; end
         else begin
            state <= S_3; x <=1'b1; end
   S_2: begin state <= S_3; x <= 1'b1; end
   S_3: if (a==1'b1) begin
            state <= S_1; x <= 1'b1; end
         else begin
            state <= S_0; x <=1'b0; end
   default: begin state <= S_0; x <= 1'b0; end
   endcase
endmodule
```

Figure 1.3

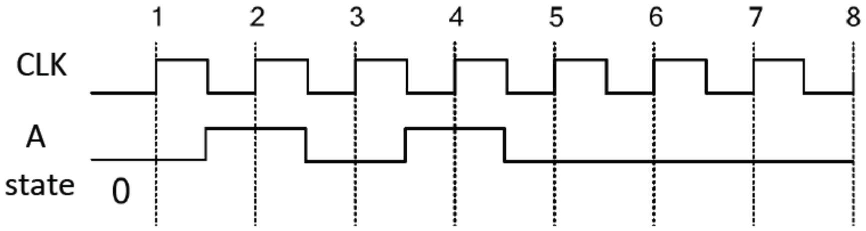


Figure 1.4

(d) A microprocessor system with a 16-bit memory address bus and an 8-bit data bus. The system consists of two banks of ROMs, ROM_1 and ROM_2, and one bank of RAM.

The two banks of ROMs are 8kB and 16kB in size and with addresses starting from 16’hE000 and 16’h8000 respectively. The RAM is 32kB in size and occupies the address space starting from 16’h0000. The input/output space (IO) occupies 32 byte addresses starting from 16’hDFE0.

(i) Draw the memory map for the microprocessor system showing the starting and ending addresses for ROM_1, ROM_2, RAM and IO.

[4]

(ii) Derive in the form of Boolean equations the address decoder circuit which produces enable signals for ROM_1, ROM_2, RAM and IO.

[4]

(e) In the circuit of *Figure 1.5*, the propagation delay of the flip-flops is t_p = 2ns, and the setup and hold time are t_s = 2 ns and t_h = 1 ns respectively. The propagation delay of the combinational circuit X is in the range of 1 ns < t_X < 3 ns. The propagation delay of the non-inverting clock buffer circuit Y is in the range of 0 ns < t_Y < 2 ns. The clock signal *CLK* is symmetrical with period T.

(i) Given the timing waveforms for signals A and CLK are as shown in Figure 1.6, draw the timing waveforms for signals B, C, D and E showing all the delay values. You may assume that all signals B, C, D and E are initially low.

[3]

(ii) Derive the inequalities for the setup times applied to the rightmost flip-flop.

[3]

(iii) Hence or otherwise, derive the maximum clock frequency for the circuit.

[2]

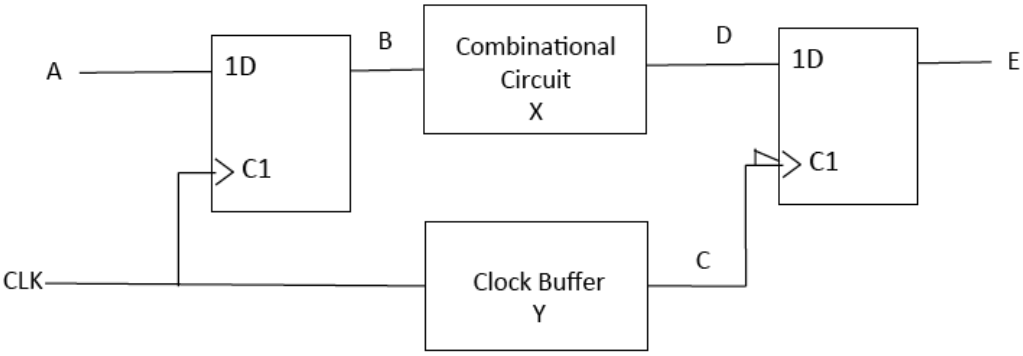


Figure 1.5

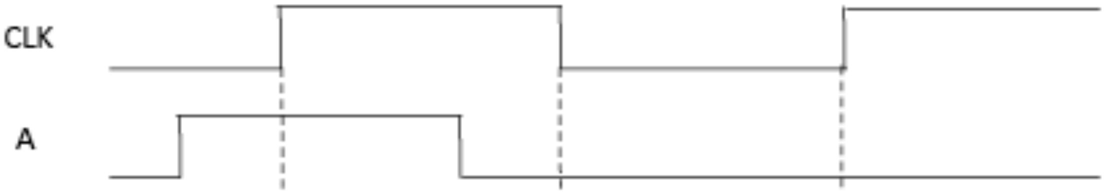


Figure 1.6

2. a) Explain briefly the principle of the successive approximation algorithm. State briefly its advantages and disadvantages when used to implement an analogue to digital converter.

[5]

b) Using the 4 x 4 unsigned multiplier shown in *Figure 2.1*, design in Verilog HDL a squaring circuit that produces an 8-bit product P which is the square value of the 4-bit unsigned input number X, i.e. $P = X^2$.

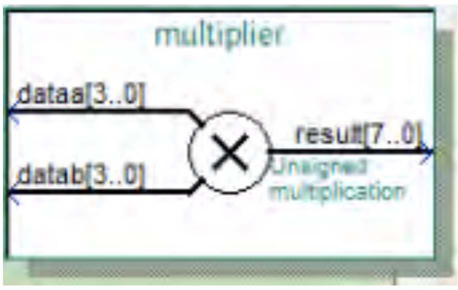
[2]

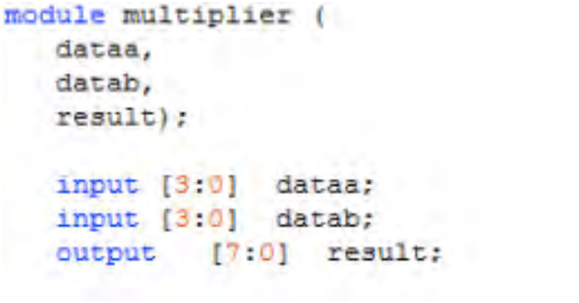
c) A digital circuit is required to compute the approximate square root of an 8-bit number using the successive approximation algorithm. The output X is the largest integer such that X^2 is less than or equal to the input Y. Using the unsigned multiplier in b), design the circuit in Verilog HDL or as a schematic diagram. (If your design is in schematic form, specify the FSM in the form of a state diagram.)

[15]

d) Demonstrate how your design works by considering how it calculates the square root of 8'h65, showing cycle by cycle the progress of the successive approximation algorithm.

[8]





```
module multiplier (
    dataa,
    datab,
    result);

    input  [3:0]  dataa;
    input  [3:0]  datab;
    output    [7:0]  result;
```

Figure 2.1

3. *Figure 3.1* shows the top-level schematic of the serial peripheral interface SPI used in the laboratory experiment to interface between the Cyclone III FPGA board and the digital to analogue converter.

   a) Specify in Verilog HDL the divide-by-50 clock circuit that produces the 1MHz internal symmetrical clock signal clk_1MHz (i.e. 1:1 mark-space ratio) from the 50MHz system clock.

   [5]

   b) *Figure 3.2* shows the state diagram of the "load detector" FSM circuit. Specify in Verilog HDL the code segment that implements this FSM.

   [10]

   c) *Figure 3.3* shows the code segment of the 16-bit data shift register circuit in Verilog HDL. Draw the digital circuit that is expected to be synthesized, using only D flip-flops, multiplexers and basic logic gates.

   [15]

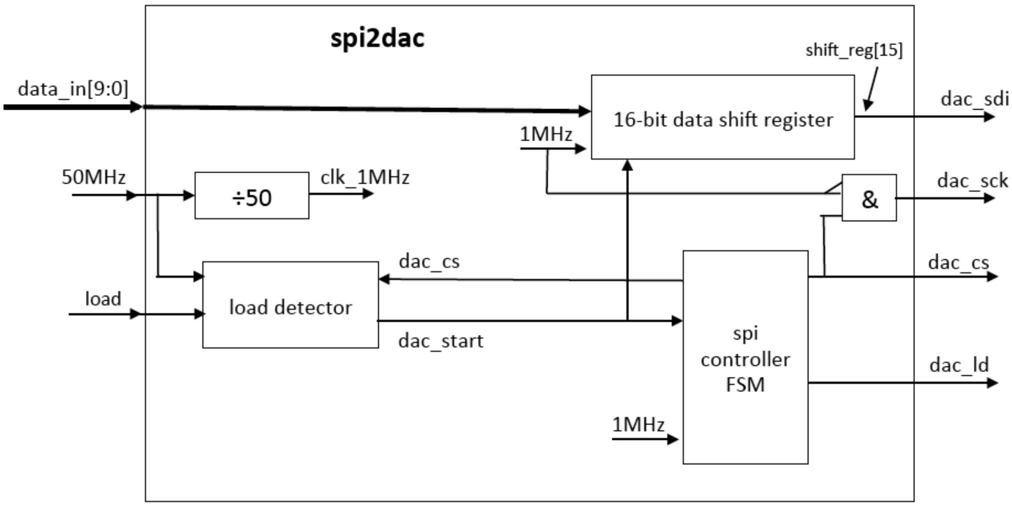


Figure 3.1

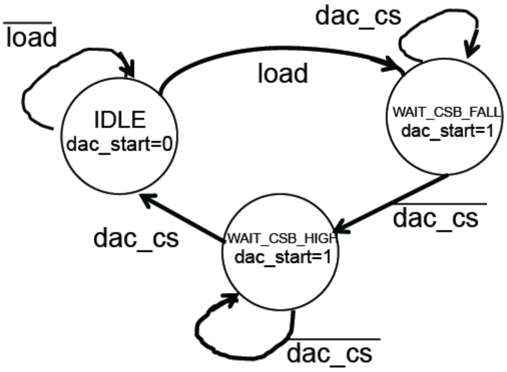
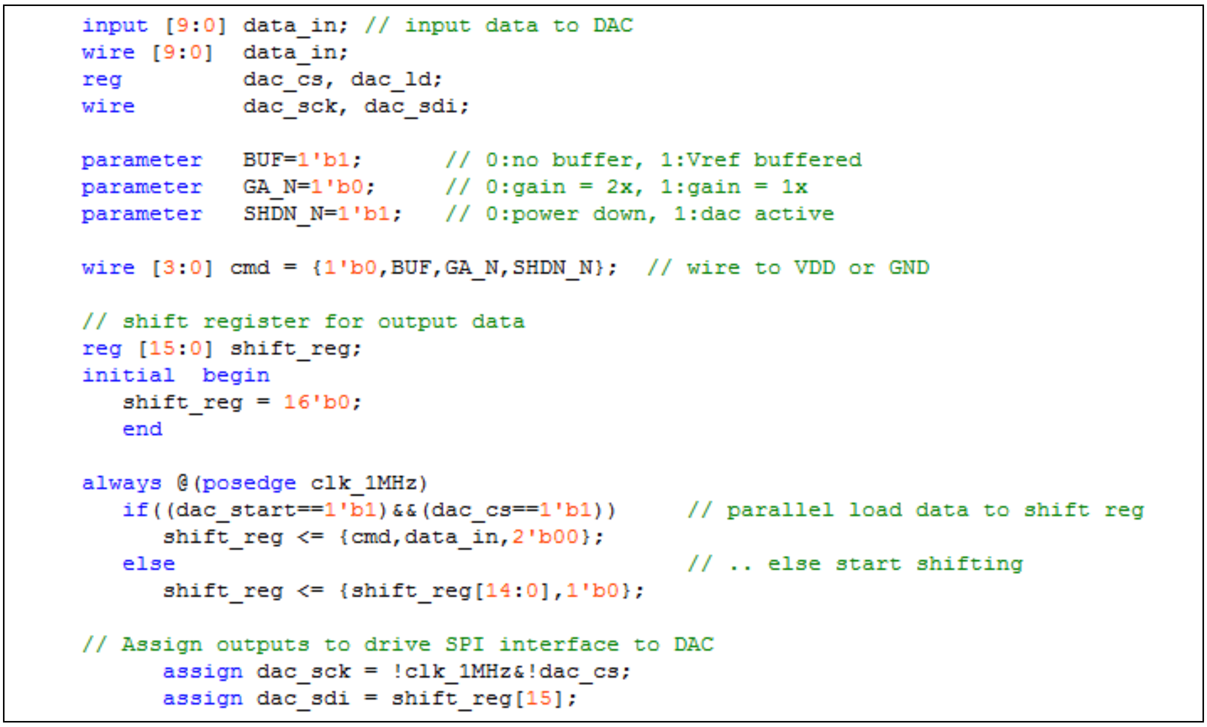


Figure 3.2



```
input [9:0] data_in; // input data to DAC
wire [9:0]  data_in;
reg         dac_cs, dac_ld;
wire        dac_sck, dac_sdi;

parameter   BUF=1'b1;      // 0:no buffer, 1:Vref buffered
parameter   GA_N=1'b0;     // 0:gain = 2x, 1:gain = 1x
parameter   SHDN_N=1'b1;   // 0:power down, 1:dac active

wire [3:0] cmd = {1'b0,BUF,GA_N,SHDN_N};  // wire to VDD or GND

// shift register for output data
reg [15:0] shift_reg;
initial  begin
   shift_reg = 16'b0;
   end

always @(posedge clk_1MHz)
   if((dac_start==1'b1)&&(dac_cs==1'b1))     // parallel load data to shift reg
      shift_reg <= {cmd,data_in,2'b00};
   else                                      // .. else start shifting
      shift_reg <= {shift_reg[14:0],1'b0};

// Assign outputs to drive SPI interface to DAC
      assign dac_sck = !clk_1MHz&!dac_cs;
      assign dac_sdi = shift_reg[15];
```

Figure 3.3

2015 Paper E2.1: Digital Electronics II- Solutions

Q1 (a) This question tests students basic understanding of FPGA architecture and how to deal with simple timing and Verilog codes.

(i) 16-bit adder requires 16 LEs to implement; each already has an in-built D-FF. Therefore total required LEs = 16.

[2]

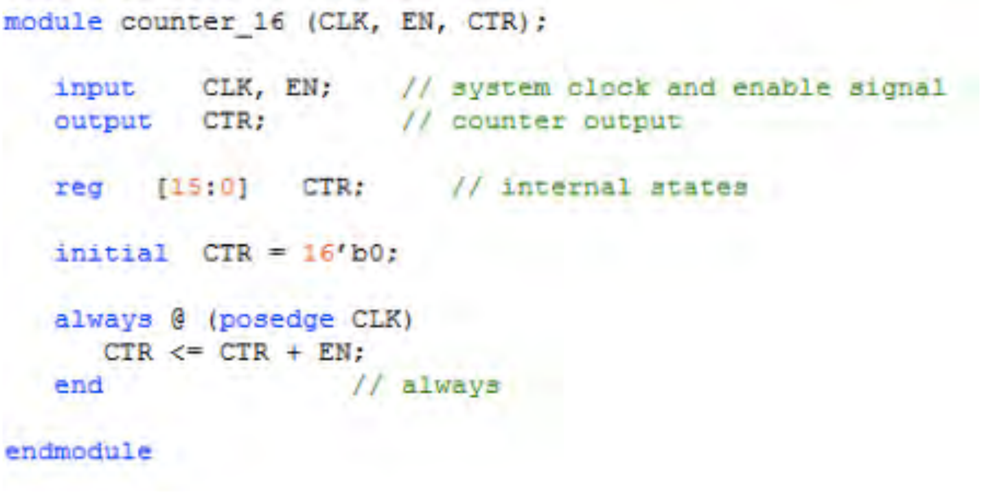
(Many students did not get this right. There were two common mistakes: 1) recognising that 16-bit counter would need 16 registers, and therefore 16 LEs were needed right away, but then the LUT resources in these LEs were not used to implement the adders, and therefore gave an answer of 32 or more. They get 1 mark for getting the 16 register = 16 LEs. 2) blindly divided 16 by 4 (because each LUT has 4 inputs), and gave the answer of 4 LEs. Answer to this question shows that many students are still very unclear about the internal hardware architecture in an FPGA LE (or CLB).)

(ii) Total adder delay = 16 x 250ps. Worst-case delay is only dictated by the FF propagation delay and the setup time here. The hold time has no relevance. Therefore td(max) = 4ns + 80ps + 100ps. Maximum operating frequency of the 16-bit counter is: 1/(4.18ns) = 239MHz.

[3]

(Only those students who got (i) correct will have a chance of full marks for this part. Those who demonstrated that they understood how to calculate the worst case frequency with their wong assumptions get 1 or 2 marks, depending on the nature of the answers.)

(iii)



```
module counter_16 (CLK, EN, CTR);

   input    CLK, EN;     // system clock and enable signal
   output   CTR;         // counter output

   reg   [15:0]   CTR;      // internal states

   initial  CTR = 16'b0;

   always @ (posedge CLK)
      CTR <= CTR + EN;
   end                // always

endmodule
```

[3]

(Most students got this right. Those who got this wrong revealed how little they understood the basics of designing digital circuits using Verilog.)

(Overall students did very well on this sub-question.)

Q1 (b) This question tests students understanding of R-2R DAC converters.

(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 Full-scale volrage / (2^N-1).

Linearity error – this is the maximum deviation of the output vs input characteristic from the linear line joining the maximum point with the minimum point.

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

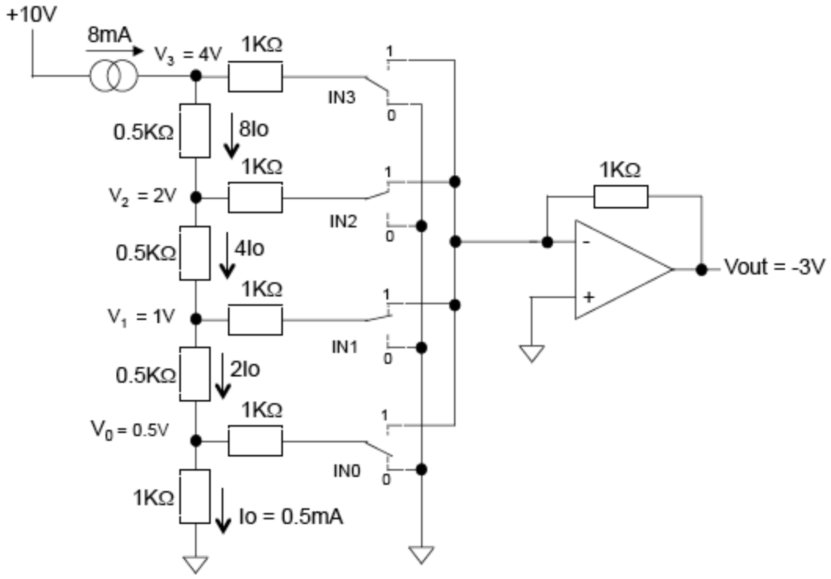
Settling time – Time taken to reach final value as input changes.

Monotonic behaviour is most important in a control loop implementation using DAC. If the characteristic is not monotonic, the gain of the system may change sign, and could then cause the system to become unstable.

[2]

(I was surprised by how poorly some students did in this “bookwork” question. The answers were straight from the notes and yet many did not even know the meaning of resolution, let alone the idea of monotonicity. This may be because I did not have a question like this in the tutorial sheets.)

(ii) The currents and voltages are shown below.



[4]

(Most students got this right. Some students are obviously weak in analogue electronics and found this question difficult. My suspicion is that they may be on the EIE course, which has considerably less analogue circuits in their syllabus.)

Q1 c) (i) In one-hot encoding, each state is identified by one bit in the state code.

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

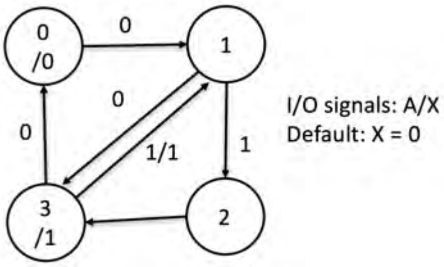
One-hot encoding is particularly suited to FPGA architecture because:

1) Generally it has less 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.
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]

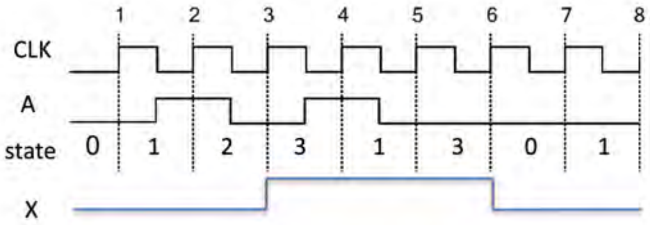
(Most students did really well in this part.)

(ii) The state diagram is:



[3]

The timing diagram is:



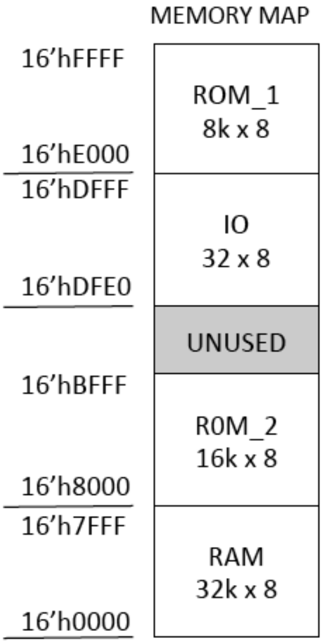
[2]

(Most students got high marks for this part of the question. The most common mistake was to draw a Moore state machine with each state having a constant output, which is wrong. Some students provided the state-transition table instead of state diagram – they loose some marks as a result. However, in Verilog code given here, X only changes on the rising edge of clock. Therefore we need to model a Mealey machine with output X' which get registed by on rising edge. Its value depends on which state is comes from.

Most students got the timing diagram right. The question did not explicitly asked for the timing diagram for the output x, but only for the variable state. Therefore the value of x shown here is not required in the answer.)

Q1 (d) This question tests students understanding memory addresses and address decoding.

(i)



[4]

(iii) Enable signals are:

EN_ROM_1 = A15&A14&A13
EN_ROM_2 = A15&/A14
EN_RAM = /A15
EN_IO = A15&A14&/A13&A12&A11&A10&A9&A8&A7&A6&A5

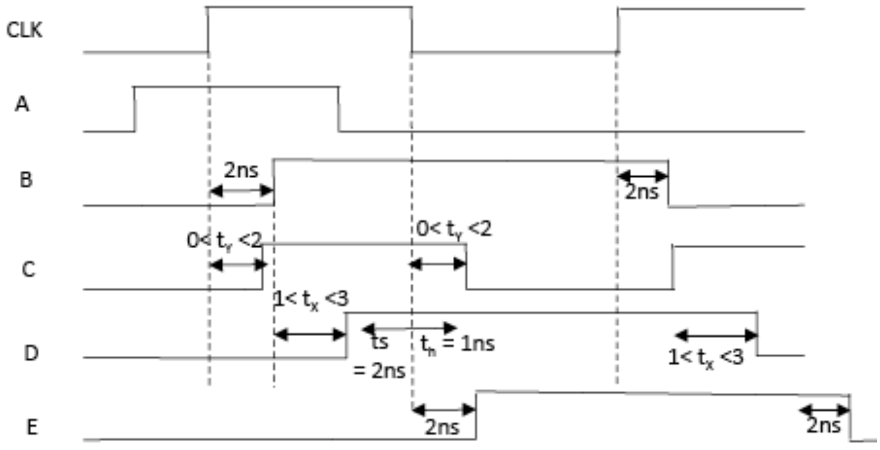
[4]

(It was pleasing to see that almost all students got this part of Q1 correctly.)

Q1 (e) This question tests student’s ability to work out setup and hold timing constraints for generic circuit.

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

(i)



[3]

(ii)

Setup inequality:

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

⇨ $2 + 3 + 2 < 0 + ½\, T$

⇨ T > 14ns

[3]

(iii)

Hence T > 14ns, fmax < 71.4MHz. [2]

(Most students answered this part of Q1 perfectly. Those who could not do this also did poorly in the other parts of the paper, showing a lack of grasp to the subject as a whole.)

(Question 1 is intended to be the ‘Mastery Question’ – it tests students basic grasp of the subject across the whole syllabus. The target average marks for the class is 30/40. The actual average was 29.7/40, which is right on target.)

Q2 This question tests student's understanding of the successive approximation algorithm, and how it could be applied to problems outside ADC.

(a) Bookwork – students are expect to provide succinct description of the successive approximation algorithm compatible with the allocated marks (equivalent to 5 to 6 minutes).

[5]

(Most students managed to answer this part of Q2 reasonably well.)

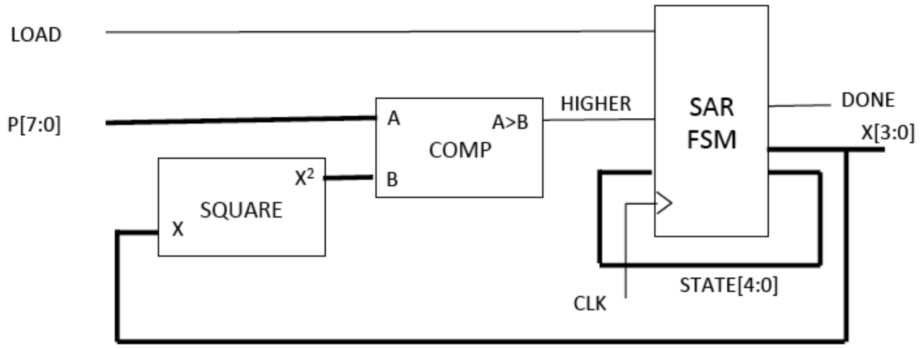
(b) This absolutely straightforward:

```
module square (x, p);
    input [3:0]        x;
    output [7:0]       p;
    multiplier MULT (x, p);
 endmodule;
```

[2]

(In spite of the explicit instruction in the question to use the multiplier block as a component, a significant proportion of students still used the '*' operator to implement the square module. However, most students got this easy part of Q2.)

(c) This is similar to the design for a SAR ADC, except that the DAC and the analogue comparator are replaced by the squaring circuit and the digital comparator respectively as shown below. The SAR FSM design is exactly the same as the for the ADC design.



[15]

(This part of the question really revealed how much a student understood the subject. Some student did not realise the link between this and the earlier parts of Q2 – strange!)

(d) The progression for evaluation square root of P = 4'h65 cycle by cycle is:

1. X[3:0] = 4'h8, X^2 = 8'h40, HIGHER = true. Therefore keep X[3] = 1.
2. X[3:0] = 4'hC, X^2 = 8'h90, HIGHER = false. Therefore reset X[2] to 0.
3. X[3:0] = 4'hA, X^2 = 8'h64, HIGHER = true. Therefore keep X[1] = 1.
4. X[3:0] = 4'hB, X^2 = 8'h79, HIGHER = false. Therefore reset X[0] to 0.
5. Final square root value is 4'hA.

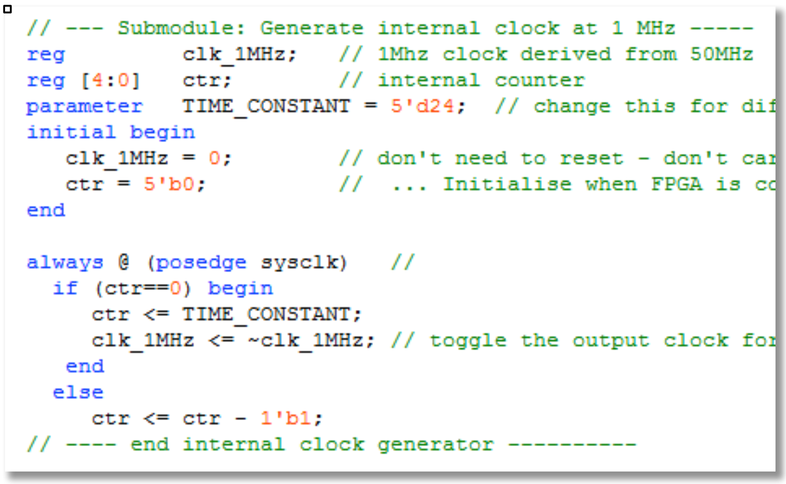
[8]

(This part of the question is easy, and does not even require part (c)! Many students got this right.)

(This question is not easy and therefore the class average mark was 10.7/30, which is lower than my target of 15/30. However, a few students got full marks, showing that this question is do-able.)

Q3 This question tests student's understanding of the serial peripheral interface, which is covered both through the lectures and the laboratory experiment.

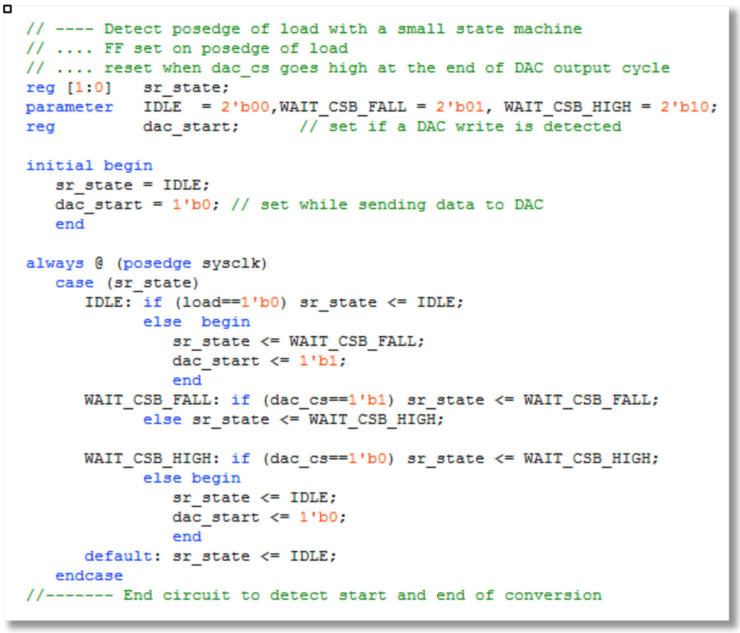
(a) This is relatively straightforward:



```
// --- Submodule: Generate internal clock at 1 MHz -----
reg          clk_1MHz;    // 1Mhz clock derived from 50MHz
reg [4:0]    ctr;         // internal counter
parameter    TIME_CONSTANT = 5'd24;  // change this for dif
initial begin
   clk_1MHz = 0;          // don't need to reset - don't car
   ctr = 5'b0;            //  ... Initialise when FPGA is co
end

always @ (posedge sysclk)   //
  if (ctr==0) begin
     ctr <= TIME_CONSTANT;
     clk_1MHz <= ~clk_1MHz; // toggle the output clock for
   end
  else
     ctr <= ctr - 1'b1;
// ---- end internal clock generator ----------
```

[5]

(Most students got this right because it is straight from the VERI experiment from the Autumn term.)

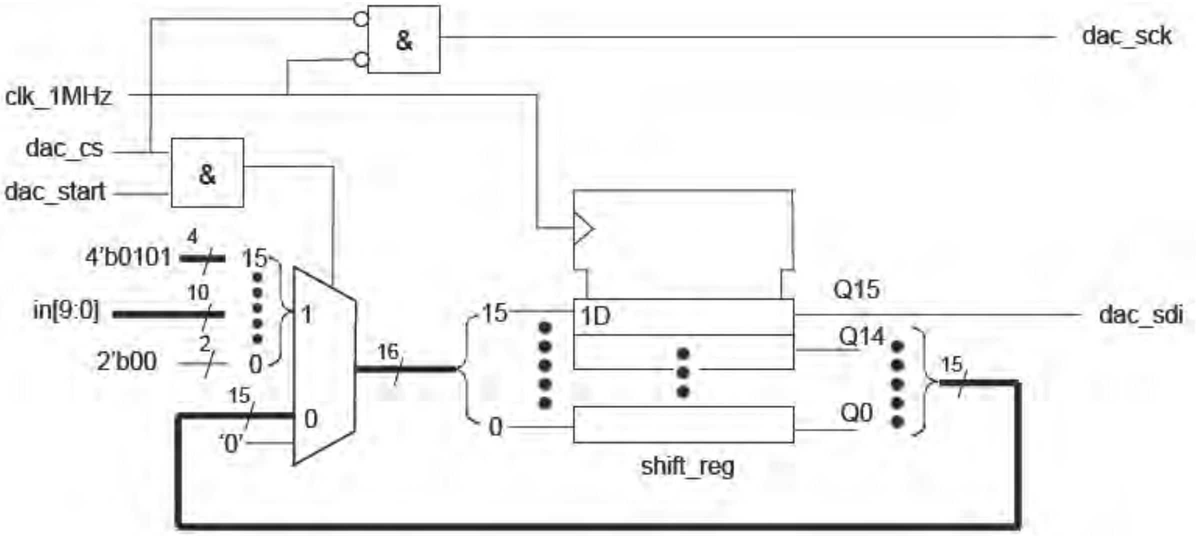


```
// ---- Detect posedge of load with a small state machine
// .... FF set on posedge of load
// .... reset when dac_cs goes high at the end of DAC output cycle
reg [1:0]   sr_state;
parameter   IDLE  = 2'b00,WAIT_CSB_FALL = 2'b01, WAIT_CSB_HIGH = 2'b10;
reg         dac_start;      // set if a DAC write is detected

initial begin
   sr_state = IDLE;
   dac_start = 1'b0; // set while sending data to DAC
   end

always @ (posedge sysclk)
   case (sr_state)
      IDLE: if (load==1'b0) sr_state <= IDLE;
            else  begin
               sr_state <= WAIT_CSB_FALL;
               dac_start <= 1'b1;
               end
      WAIT_CSB_FALL: if (dac_cs==1'b1) sr_state <= WAIT_CSB_FALL;
            else sr_state <= WAIT_CSB_HIGH;

      WAIT_CSB_HIGH: if (dac_cs==1'b0) sr_state <= WAIT_CSB_HIGH;
            else begin
               sr_state <= IDLE;
               dac_start <= 1'b0;
               end
      default: sr_state <= IDLE;
   endcase
//------- End circuit to detect start and end of conversion
```

[10]

(Most students did very well in this part of Q3. They obviously can design FSM in Verilog.)

(c) This part of the question tests student's understanding of how Verilog HDL is mapped to digital circuits.



[15]

(This part of Q3 revealed those who really could relate Verilog specification to hardware implementations. Around half the students got this part almost perfectly.)

(It is clear that those who revised both the notes AND the experiment did well in Q3, and those who did not take the experiment seriously struggled with this question. Overall they did Q3 better than Q2, with a class average of 17/30 as compared with the target of 15/30.)

This year's paper is near perfect from the point of view of mark distribution. The overall average paper average was 58%, very close to the ideal average of 60%. Q1, the "Mastery" question scored an average of 29/40 (target 30/40). Q2 is the hardest with an average of 10.6/30 (target 15/30). Q3 is easier, with an average of 17.9/30 (target 15/30).