IMPERIAL COLLEGE LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
EXAMINATIONS 2015

EEE/EIE PART II: MEng, BEng and ACGI

DIGITAL ELECTRONICS 2

Monday, 1 June 2:00 pm

Time allowed: 2:00 hours

Corrected Copy

There are THREE questions on this paper.

Answer ALL questions.
Q1 carries 40% of the marks. Questions 2 and 3 carry equal marks (30% each).

① 2:30 correction to Fig 1.3

Any special instructions for invigilators and information for candidates are on page 1.

Examiners responsible First Marker(s) : P.Y.K. Cheung
Second Marker(s) : W. Dai

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]

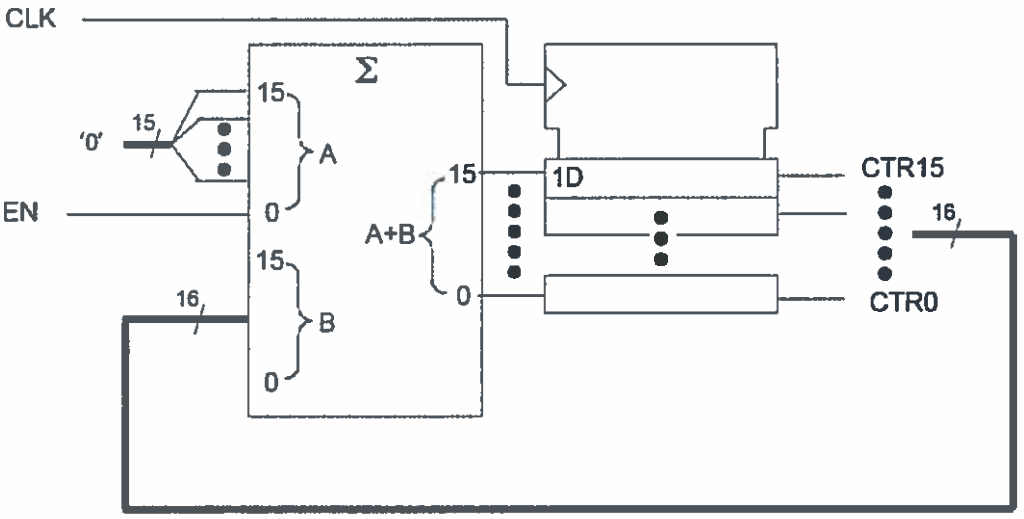


Figure 1.1

(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]

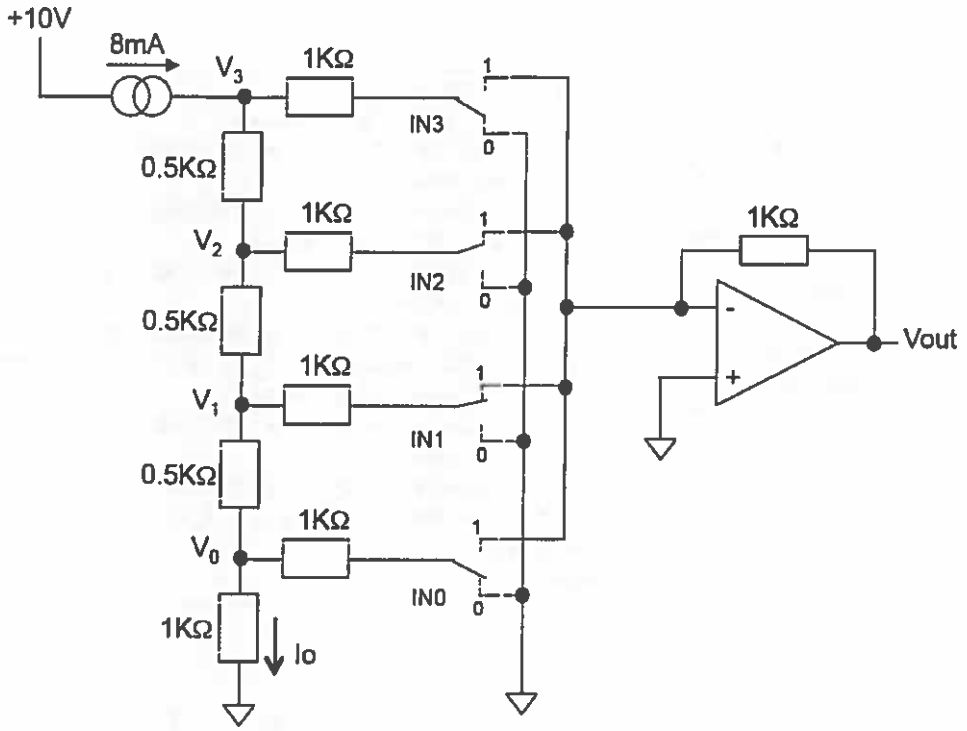


Figure 1.2

(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

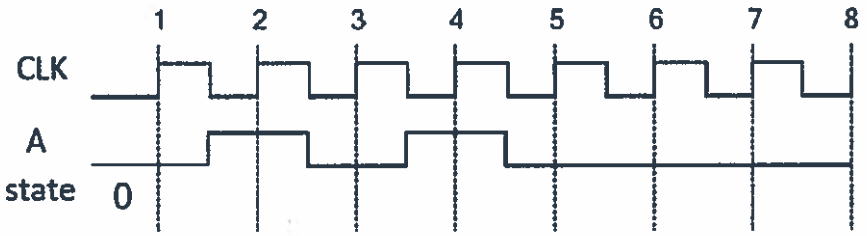


Figure 1.4

(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]

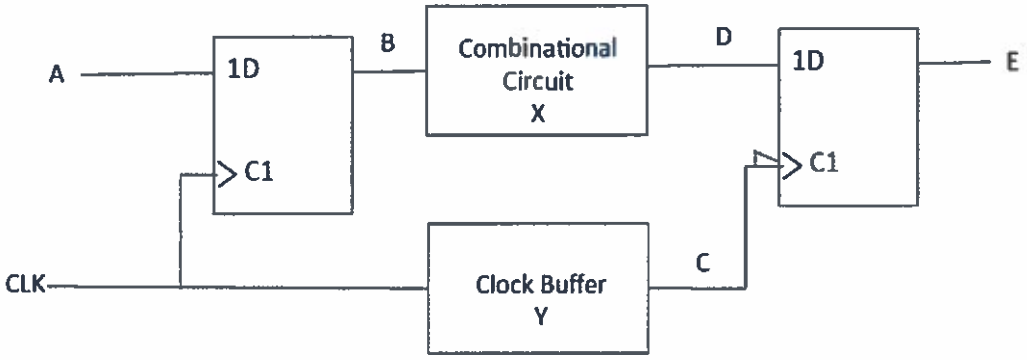


Figure 1.5

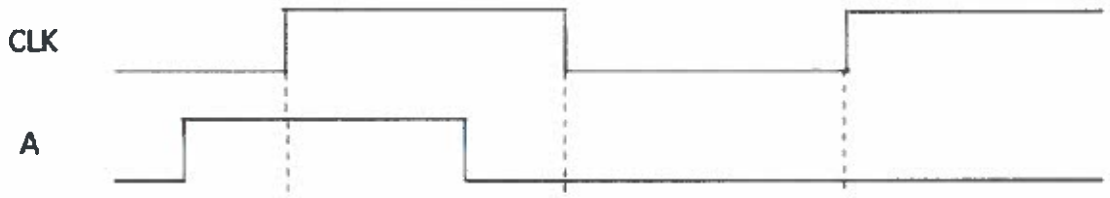


Figure 1.6

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. $P = X^2$.

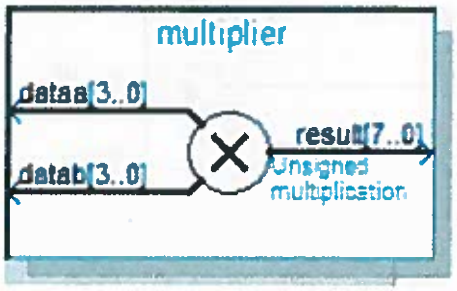
[2]

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]

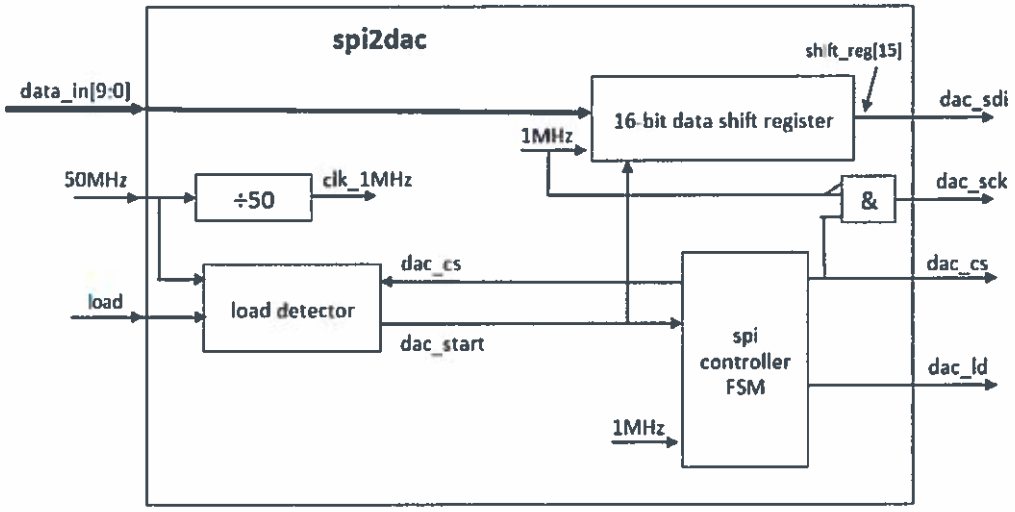


Figure 3.1

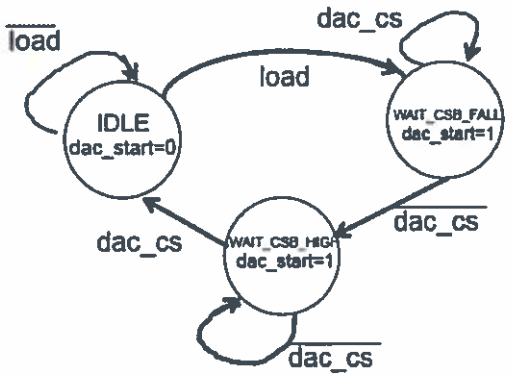


Figure 3.2

```
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