2014 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 binary counter with a clock signal CLK, a synchronous reset input R, and a synchronous select input S. The select input S is low if the counter is a 10-bit counter, and high if it is a 12-bit counter. The counter has one output signal TO, which goes high for one clock cycle when the counter reaches a value of 1023 or 4095 if S is 0 or 1 respectively. Specify this counter circuit in Verilog HDL so that it can be synthesized. Estimate and justify the number of Logic Elements (LEs) required to implement this design on a Cyclone III FPGA.

[8]

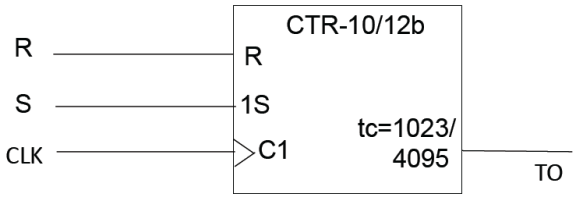


Figure 1.1

(b) Explain the principle of operation of a 12-bit pulse-width modulation (PWM) digital-to-analogue converter (DAC). Design in Verilog HDL a PWM DAC with an interface as defined in *Figure 1.2*.

[8]

```
module pwm_dac (clk, data_in, pwm_out);

   input          clk;        // system clock
   input [11:0]   data_in;    // input data for conversion
   output         pwm_out;    // PWM output
```

Figure 1.2

(c) *Figure 1.3* shows the timing diagram of a shift register based control circuit with a clock signal CLK and a trigger input signal TRIGGER, which changes on the falling edge of CLK and is a pulse lasting for one period of CLK. Design in schematic or Boolean equations the control circuit that generates output signals X and Y as shown in the timing diagram using shift registers and logic gates.

[8]

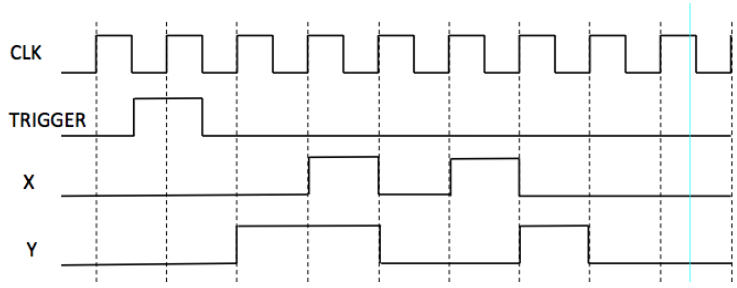


Figure 1.3

(d) *Figure 1.4* shows a circuit with two D-flipflops FF1 and FF2 with setup and hold times of 3 ns and 2 ns respectively, and a clock-to-Q output delay of 2 ns. The clock signal CLK has a 1:1 mark-space ratio. The datapath is driven by logic_A, which has a propagation delay between 2 ns and 5 ns. The clock path is driven by logic_B, which has a propagation delay between 1 ns and 10 ns. Derive the maximum frequency of the signal CLK for reliable operation of this circuit.

[8]

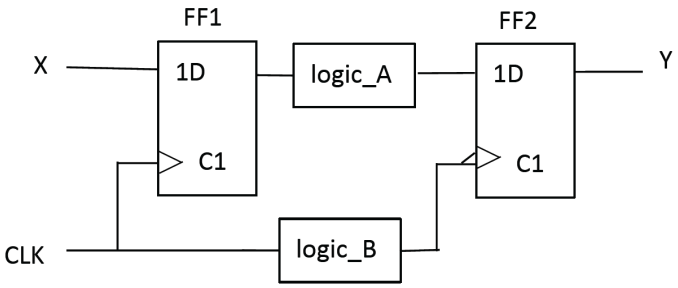


Figure 1.4

(e) A microprocessor system with an 18-bit memory address bus has a memory map as shown in *Figure 1.5*. The system consists of two banks of RAMs, RAM_1 and RAM_2, one bank of ROM, and a 32-location space for input and output. The two banks of RAMs are respectively 32k and 16k in size, and occupy consecutive address spaces starting from address 0. The ROM is 64k in size and occupies the address space starting from 18'h10000. The input/output space starts from 18'h3FF80. Design in the form of Boolean equations the address decoder circuit which produces the RAM_1, RAM_2, ROM_1 and INPUT_OUTPUT enable signals.

[8]

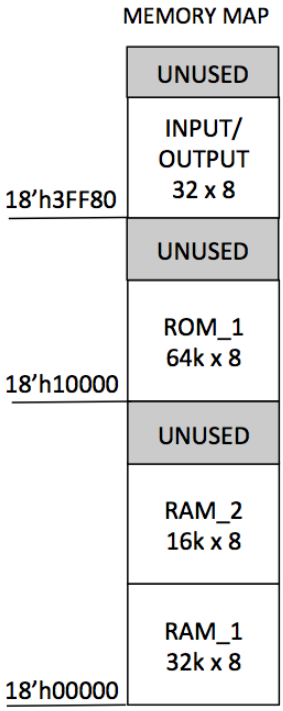


Figure 1.5

2. A finite state machine controlled by the clock signal CLK has two inputs P and Q, and two outputs X and Y. All signal transitions occur shortly after the rising edge of CLK. The signal P contains high pulses lasting for exactly one clock cycle that are separated by one or more clock cycles. The signal Q is similar to P, but pulses on P and Q never occur together, i.e. P and Q are never high simultaneously. *Figure 2.1* shows a typical sequence of the input signals P and Q.

   The finite state machine produces the output X, which detects two or more consecutive pulses occurring on P without being cancelled by a pulse on Q. For example, referring to *Figure 2.1*, pulse "a" on P is cancelled by pulse "b" on Q, and X stays low. However, pulses "c" and "d" are two consecutive pulses on P. Therefore X goes high at the next rising edge of CLK, producing the output pulse "e" on X. Pulse "g" on Q causes X to reset to zero on the next rising edge of CLK. Similarly Y goes high whenever two or more pulses arrive on Q and is not cancelled by a pulse on P. For example, pulses "g" and "h" cause Y to go high at "j". Pulse "i" on P resets Y to zero. Pulses "k" and "m" on Q cause Y to go high at n, which is then reset by pulse "o" on P.

   (a) Draw a state diagram for a finite state machine that implements the above specification. Remember that P and Q are never high together.

   [15]

   (b) Specify the design of your finite state machine in Verilog HDL using one-hot state encoding.

   [15]

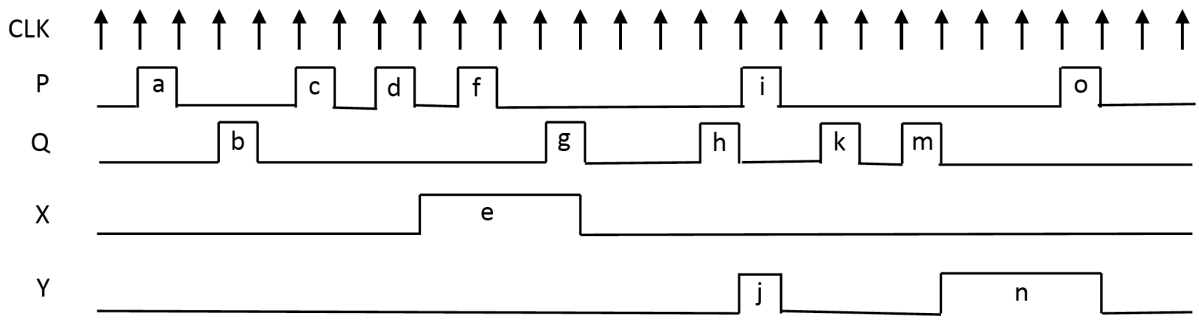


Figure 2.1

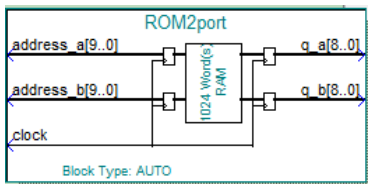
3. A dual-port 1024 x 9-bit ROM shown in *Figure 3.1* is used to store the coefficient of one cycle of a sinewave in 2's complement form. It is required to produce a pair of sinusoidal signals exactly $\pi/2$ radians apart (they are known as quadrature signals).

   a) Design in block diagram form a circuit that produces these quadrature signals with the signal frequency as close to 1kHz as possible. You may assume that a clock signal at 50MHz is available and the dual-port ROM is already provided.

   [15]

   b) Design in Verilog HDL a module to produce two 9-bit output signals *sine_sig* and *cosine_sig* which are the quadrature signals at approximately 1kHz. State to 4 significant digits the frequency of your outputs.

   [15]




```
module ROM2port (
   address_a,
   address_b,
   clock,
   q_a,
   q_b);

   input [9:0]  address_a;
   input [9:0]  address_b;
   input   clock;
   output   [8:0]  q_a;
   output   [8:0]  q_b;
```

Figure 3.1

2014 Paper E2.1: Digital Electronics II- Solutions

1. (a) This question tests students ability to write Verilog code for a simple counter.

```
module counter_10_12 (clk, r, s, to);

   input          clk;        // system clock
   input          r;          // synchronous reset input
   input          s;          // 0 - select 10-bit; 1 - select 12-bit counter
   output         to;         // timeout signal - goes high for one clock cycle

   reg [11:0]     count;      // internal 12-bit counter

   initial count = 12'b0;

   always @ (posedge clk)
      if (r == 1'b1)
         count = 12'b0;
      else begin
         count <= count + 1'b1;
         to <= 1'b0;          // timeout normally reset
         if (s==1'b0) begin      // in 10-bit mode
            if (count==12'd1023) begin
               count <= 12'b0;
               to <= 1'b1;
               end
            else                 // in 12-bit mode
               if (count==12'd4095)
                  to <= 1'b1;
         end
   end         // always

endmodule
```

This will need 12 LEs for the 12-bit counter. It also needs to detect 12'd1023 and 12'd4095. This requires 13-bit input (12 bits for counter and 1 bit for S), and two outputs. The minimum it would need is 5 LE's (for example, detect lower 10-bit as 1, this needs 4 LE's, and one more to detect the other 2 bits for 12-bit mode). However, anything between 5 and 10 would be acceptable. So accept an answer from 17 to 22 LEs.

[8]

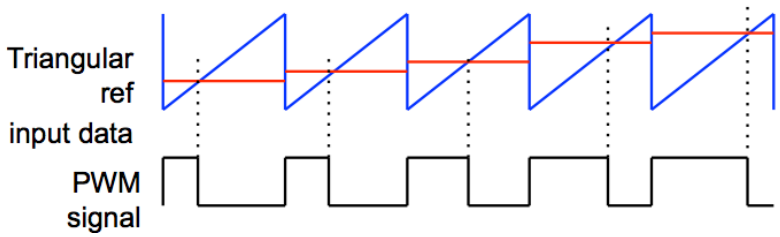
(This should have been a very easy question, particularly in light of the laboratory experiment. A small number of students obviously did not do, or understand the experiment, and manifested clear ignorance. Some students used TWO separate counters, which would loose some marks. A fairly common mistake is that the counter does not get reset to either 1023 or 4095 when the terminal count is reached, demonstrated incomplete understanding on how such a design works. But on the whole, the majority got at least 4/8 from this question.

A very common error is the number of LE required. Most students had no clue about the mapping from the Verilog description to hardware. Many said it takes 3 LEs because each LE has four inputs, and 12-bit divide by four is three. This shows complete ignorance about exactly what a LE consists of. A very small number used 12 LEs because a 12-bit counter needs 12 registers, and each LE only has one register. They loose only one mark for ignoring the LEs required to decode the terminal counter values (i.e. detect when counter_value = 1023 or 4095 for up counter, or for down counter, counter_value = 0).

Next year, I will make sure that everyone understands how such counters are mapped to FPGA circuits.)

(b) Bookwork. This question tests student's understanding of PWM DAC which has been covered in the lectures.

Generate a triangular signal (in the form of a 12-bit counter) and compare the input value data_in to that of the counter value. Set pwm_out to be high if the counter value is lower than data_in, otherwise set it to low. The DAC output is the lowpass filtered version of the PWM signal.



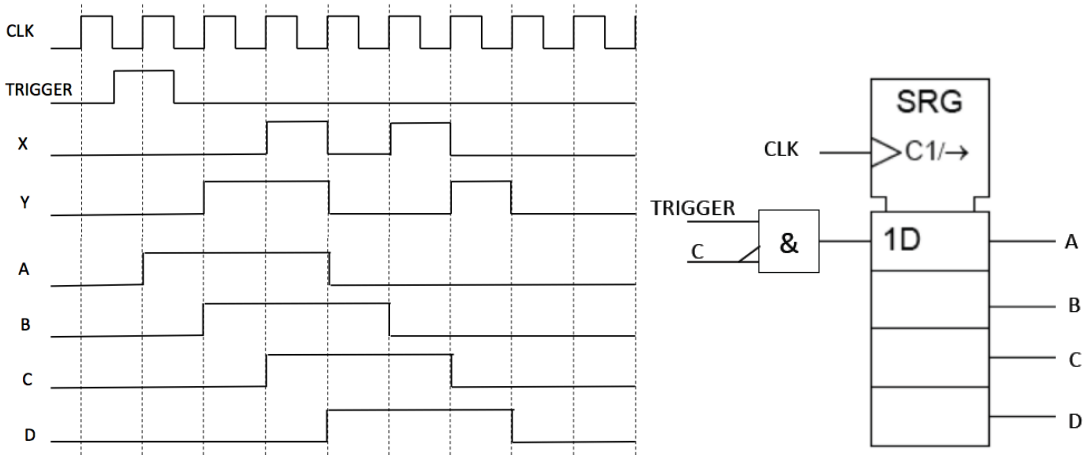

```
module pwm_dac (clk, data_in, pwm_out);

   input          clk;        // system clock
   input [11:0]   data_in;    // input data for conversion
   output         pwm_out;    // PWM output

   reg [11:0]     count;      // internal 12-bit counter
   reg            pwm_out;

   initial count = 12'b0;

   always @ (posedge clk) begin
      count <= count + 1'b1;
      if (count > data_in)
         pwm_out <= 1'b0;
      else
         pwm_out <= 1'b1;
      end

endmodule
```

[8]

(This question has caught out many students who missed the two slides in the lecture notes that describe the PWM converter. The first part is very much bookwork, and the second part is very simple if the idea is clearly understood. However, a significant number of students had no clue as to what a PWM D-A converter is, or how this could be achieved. Those who had revised that part of the notes got mostly full marks.)

(c) This question tests student's ability to use shift registers to produce control signals for digital systems.

Use four stage shift register with outputs A, B, C and D as shown.



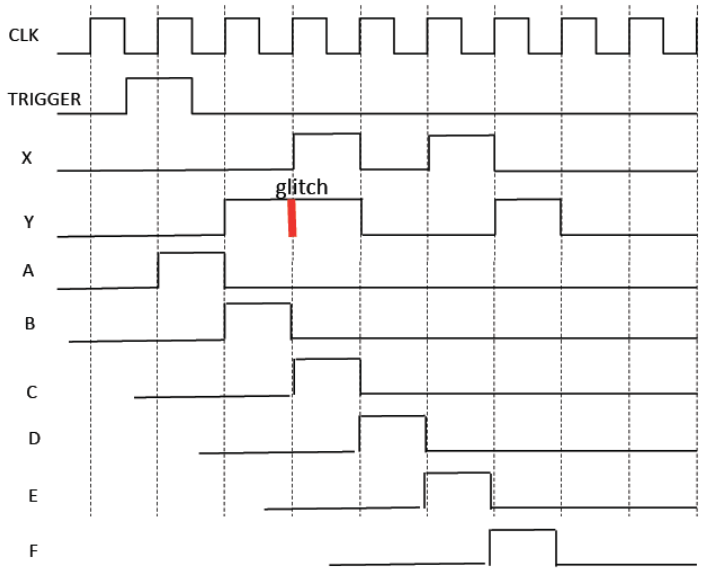
Then X = A & C + ~B & C

Y = A & B + ~C & D

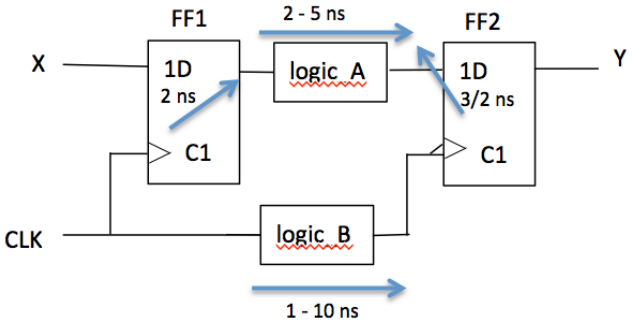
[8]

(This is a relatively straight forward question but many students got caught out because they did not revise the Chapter on shift-register based control circuit. Here the shift register has the trigger signal as shift input through a AND gate. So when C is 0, TRIGGER will cause a 1 to be shifted to A on the next rising edge of CLK. When C goes high, a zero is shifted into A, thus resetting the sequence.

Many students took the easy route, which is almost OK. They use 6 stage shift register with outputs A to F. The timing diagram is as shown below. Then X = B + E, and Y = B + C + F. Unfortunately this design not only uses more stage of SR, but also will give a glitch in Y. This answer will have 2 marks deduced.)



(d) This question tests student's ability to work out digital circuit timing constraints.



Setup time constraint:

tc-q(max) + logic_A(max) + t_setup < ½ T + logic_B(min)

2 + 5 + 3 < ½ T + 1, therefore T > 18ns and Fmax (setup) < 55.56MHz

Hold time constraint:

T + tc-q(min) + logic_A(min) > ½ T + logic_B(max) + t_hold, therefore T > 16ns. Hold time is never violated.

[8]

(Most students got full marks for this question. A few students did not realised that the two FFs are clocked by opposite phase of CLK. Therefore their equations had T instead of ½ T.)

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

The address ranges for the four spaces are:

RAM_1: 18'h00000 to 18'h07FFF

RAM_2: 18'h08000 to 18'h0BFFF

ROM_1: 18'h10000 to 18'h1FFFF

I/O: 18'h3FF80 to 18'h3FF9F

Therefore RAM_1 CS = ~A17 & ~A16 & ~A15

RAM_2 CS = ~A17 & ~A16 & A15 &~A14

ROM_1 CS = ~A17 & A16

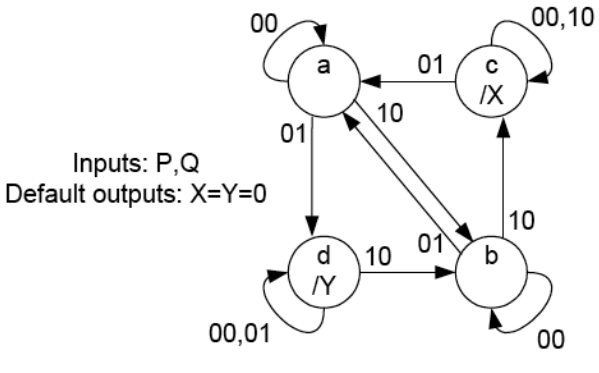
I/O CS = A17 & A16 & A15 & A14 & A13 & A12 & A11 & A9 & A8 & A7 & A6 &A5

[8]

(I was surprised how many students did not get full marks from this question. A significant number of students could not handle the fact that 18 bit address is not multiple of 4-bits, and therefore the FIVE hex-digits used here does not go from A19 – A0, but A17 – A0! In other words, the most significant digital only goes from 0 to 3. Those students making this error, but able to design the decode logic from A19 down still got most of the marks. Some students could not do the I/O address decoding, but OK with the RAM/ROM decoding.

A very large number of student got full marks in this question.)

2. This question tests student's ability to design a reasonably complicated FSM.



(a)

[15]

(b)

```
module  FSM_detector (CLK, P, Q, X, Y);

   input    CLK;     // clock input - all transitions happens shortly after rising edge
   input    P,Q;     // FSM input signals
   output   X,Y;     // FSM output signals

   parameter   STATE_a = 4'b0001, STATE_b = 4'b0010, STATE_c = 4'b0100, STATE_d = 4'b1000;

   reg [3:0]   state;

   initial begin
      state = STATE_a;
      X = 1'b0;
      Y = 1'b0;
      end
   wire [1:0]  in;

   assign in = {P,Q};

   always @ (posedge CLK)
      case (state)
         STATE_a: case (in)
                     2'b00:  state <= STATE_a;
                     2'b01:  state <= STATE_d;
                     2'b10:  state <= STATE_b;
                     default: state <= STATE_a;
                  endcase
         STATE_b: case (in)
                     2'b00:  state <= STATE_b;
                     2'b01:  state <= STATE_a;
                     2'b10:  state <= STATE_c;
                     default: state <= STATE_b;
                  endcase
         STATE_c: case (in)
                     2'b00:  state <= STATE_c;
                     2'b01:  state <= STATE_a;
                     2'b10:  state <= STATE_c;
                     default: state <= STATE_c;
                  endcase
         STATE_d: case (in)
                     2'b00:  state <= STATE_d;
                     2'b01:  state <= STATE_d;
                     2'b10:  state <= STATE_b;
                     default: state <= STATE_d;
                  endcase
         default:    ;
         endcase
```

```
   always @ (state)
      case (state)
         STATE_a: begin X = 0; Y = 0; end
         STATE_b: begin X = 1; Y = 0; end
         STATE_c: begin X = 0; Y = 0; end
         STATE_d: begin X = 0; Y = 1; end
         default: begin X = 0; Y = 0; end
      endcase

endmodule
```

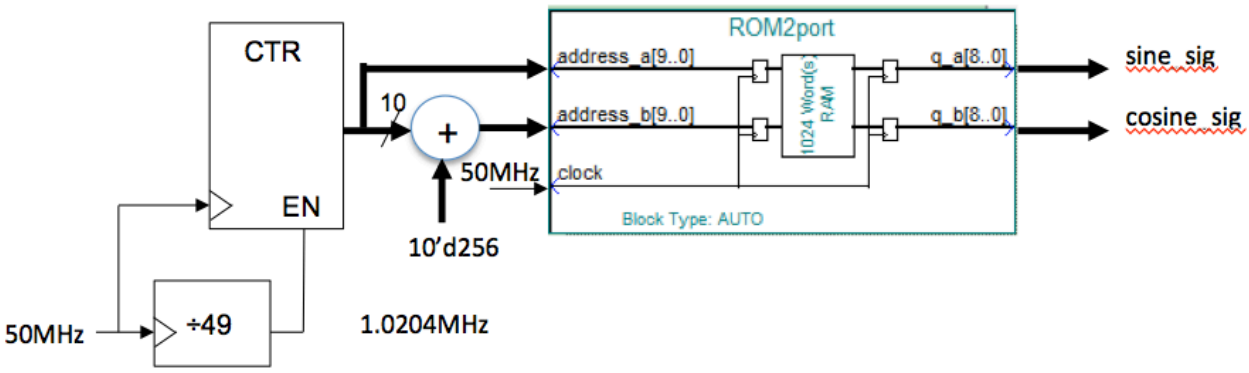
[15]

(Most students made good progress with this question. A large number create an unnecessary idle state which the FSM leaves immediately when either P or Q is asserted, but is never entered again. It is minor and only loose 2 mark. However, some students created extra states in between, which will loose them a few more marks because that could result in missing P or Q assertions (while in these extra unnecessary states).

Many students had incomplete specification of the FSM in Verilog (because P and Q are never high together). While the logic of the Verilog description is correct, as stated in my lectures many times, the synthesis software will NOT know that P & Q are never high together. With such an incomplete specification, the systems will insert extra latches to make sure that the previous outputs are maintained for these incompletely specified inputs. Many students did not have the default conditions specified, which would overcome this problem.

Overall, the class did this question really well, and the average marks for Q2 is higher than expected.)

3. (a)



[15]

(b)

```
module   quadrature_gen (CLK50, sine_sig, cosine_sig);

   input          CLK50;          // 50 MHz clock
   output [8:0]   sine_sig;       // sinewave signal
   output [8:0]   cosine_sig;     // cosinewave signal

// ---- clock divider
   reg [5:0]      clk_ctr;        // count clock cycles
   reg            time_out;       // goes high for 1 cycle every 49 clk cycles
   initial        clk_ctr = 6'b0;
   parameter      tc = 48;        // count from 0 to 48
   always @ (posedge CLK50)
      if (clk_ctr==0) begin
         time_out <= 1'b1;
         clk_ctr <= tc;
         end
      else begin
         time_out <= 1'b0;
         clk_ctr <= clk_ctr + 1'b1;
         end
// ----  end of clock divider

// ---- address counter ----

   reg [9:0]   address_a;           // address counter for sine_sig
   reg [9:0]   address_b;           // address counter for cosine_sig
   initial     address_a = 10'b0;

   always @ (posedge CLK50)
      if (time_out == 1'b1) begin
         address_a <= address_a + 1'b1;
         address_b <= address_a + 10'd246;
      end

   ROM2port sine_rom (address_a, address_b, CLK50, sine_sig, cosines_sig);

endmodule
```

fout = fsamp/1024 = (50MHz/49) / 1024 = 996.5 HZ.

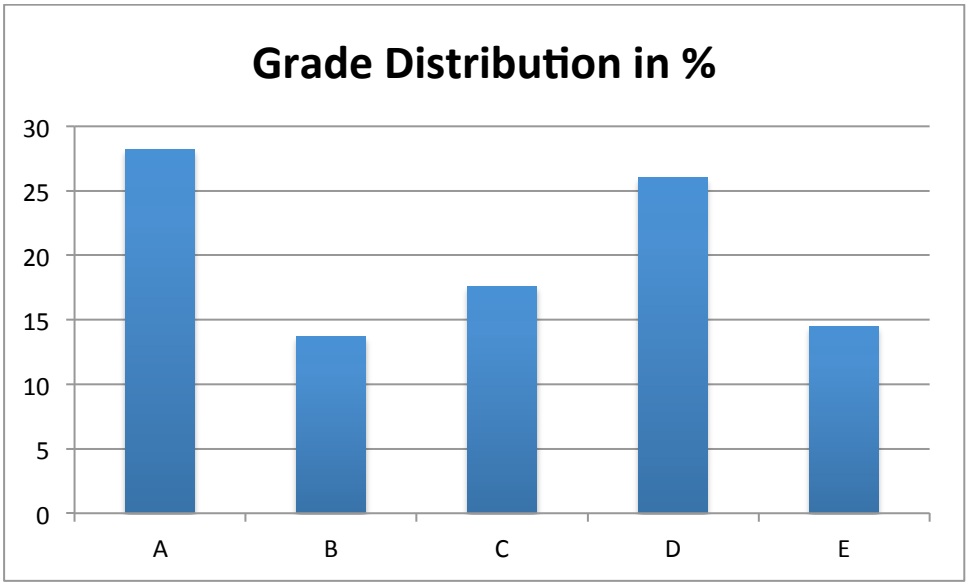
[15]

(Those students who really did the experiment until the last compulsory exercise (i.e. echo simulator) did very well on this question. I was disappointed that many students have already forgot the experiment and had no clue as to what to do with this question. Those who can do it, generally scored close to full marks. Therefore the marks for this question is very bi-modal.)

GENERAL REMARKS

I was disappointed with the answers I read for this paper. The class average was 55.4%, which is just within the acceptable nominal range of 55% to 65%.

The overall grade distribution was:



28% A grade and 14.5% E grade (failed) is just within our nominal acceptable bounds.