# COE 405, Term 152

## Design & Modeling of Digital Systems

## HW# 6 Solution

## Due date: Sunday, April 17

**Q.1.** It is required to design a circuit that computes the average, maximum and minimum of a number of scores N (assuming $0<N\leq15$), each score with a value in the range [0,15]. Once a user presses a Start button, the number of scores N will be entered. Then, scores will be entered in subsequent clock cycles one score at a time. Once the circuit finishes computation, it will assert a Done signal and will generate the average, maximum and minimum scores. The average will be shown as an integer number resulting from dividing the sum by N with rounding the result to the nearest integer. The Done signal will remain asserted unless the user presses a Reset button.

**(i)** Develop an ASMD chart for the circuit.

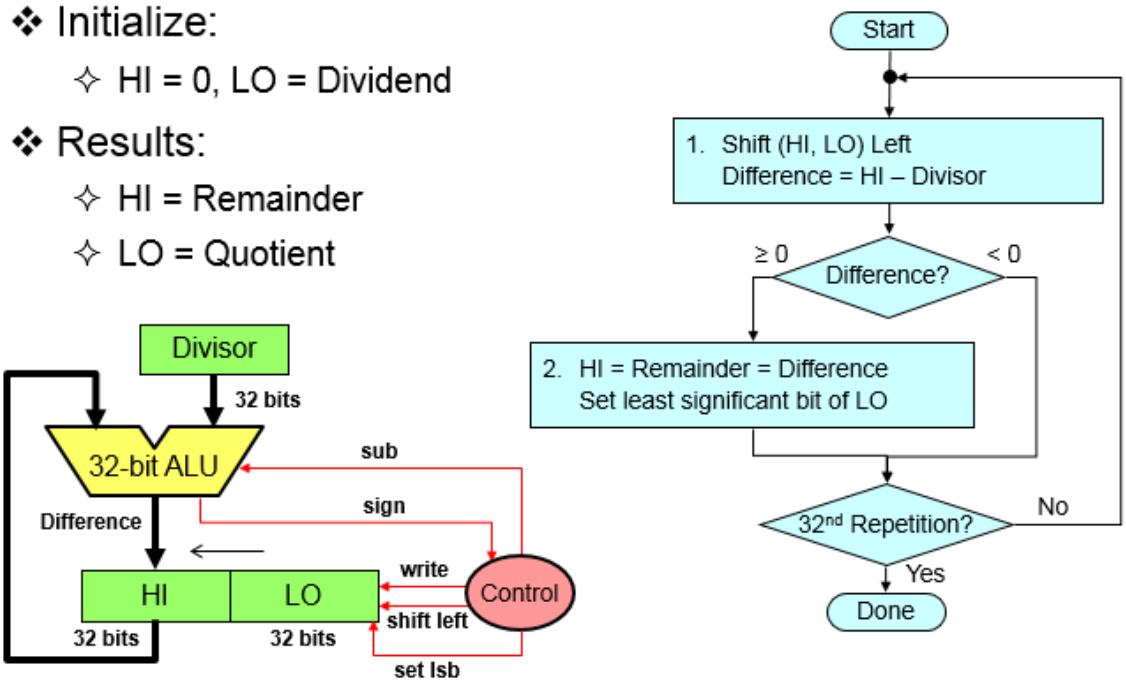
**(ii)** Show the design of your data path unit. Write a Verilog module to model your data path.

**(iii)** Write a Behavioral Verilog module to model the ASMD chart of your circuit.

**(iv)** Write a test bench to verify the correct functionality of your circuit. Show snapshots of your simulation to demonstrate its correctness.

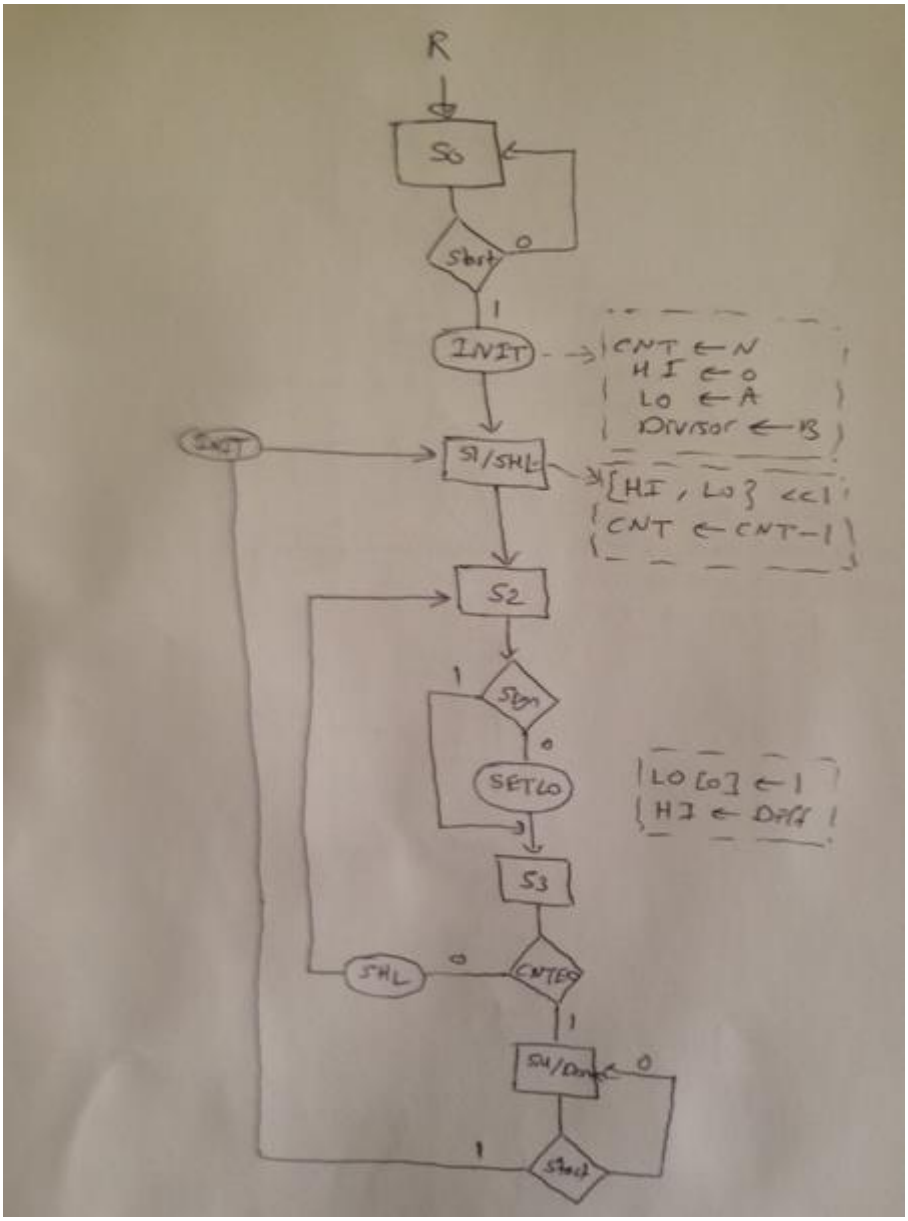
**(v)** Implement your circuit on FPGA and demonstrate its correct functionality. Include a link for a video snapshot to demonstrate correct functionality of your circuit on FPGA.

First, we will design an unsigned divider to be used for computing the average after computing the sum based on the following algorithm for unsigned division:




For our problem, the maximum value for the dividend is 15*15=225. Thus, we will use an 8-bit divider where all the registers i.e., HI, LO, Divisor are 8-bits. The Quotient and Remainder will be taken from the least significant 4-bits of LO and HI registers, respectively. Note that it is possible to optimize the design by using only 5 bits for HI and Divisor and subtractor for our example.

The ASMD chart for the unsigned N-bit divider is given below:



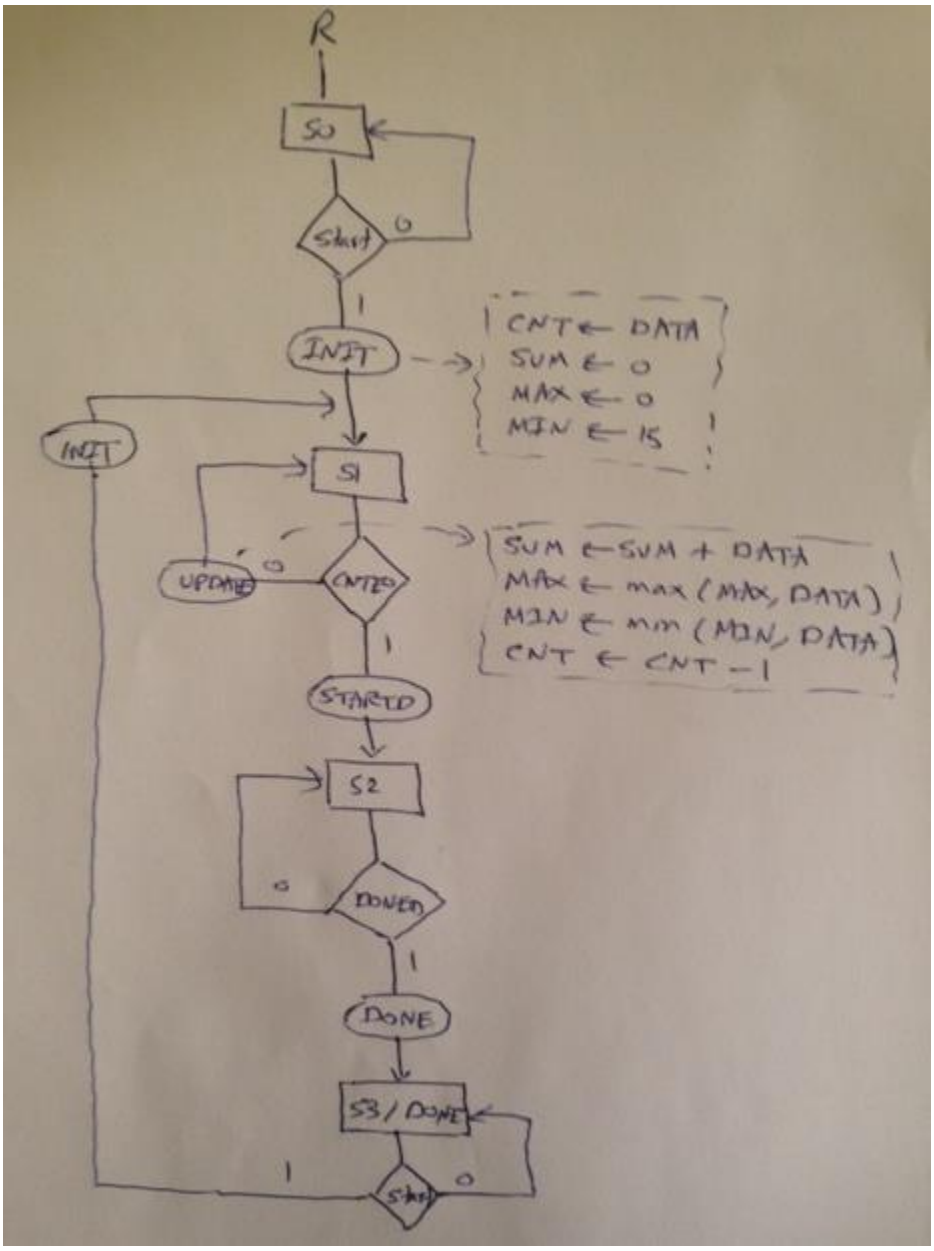

The Verilog modules for the divider are given below:

```
module Divider #(parameter N=8)(output [N/2-1:0] Q, R, output DONE,
input [N-1:0] A, B, input Start, Reset, CLK);

wire [N-1:0] HI, LO;
assign Q = LO[N/2-1:0];
```

```
assign R = HI[N/2-1:0];


Divider_DPU #(N) M1 (HI, LO, Sign, CNTEQ0, A, B, SHL, INIT, SETL0,
CLK);

Divider_CU M2 (INIT, SHL, SETL0, DONE, Start, Sign, CNTEQ0, Reset,
CLK);

endmodule



module Divider_DPU #(parameter N=8) (output reg [N-1:0] HI, LO, output
Sign, CNTEQ0, input [N-1:0] A, B, input SHL, INIT, SETL0, CLK);



reg [N-1:0] Divisor;
reg [log2(N):0] CNT;
wire [N-1:0] Diff;

assign Diff = HI - Divisor;
assign CNTEQ0 = ~| CNT;
assign Sign = Diff[N-1];


always @(posedge CLK)
if (INIT) begin
      CNT <= N;
      Divisor <= B;
      HI <= 0;
      LO <= A;
end
else if (SHL) begin
            HI <= {HI[N-2:0], LO[N-1]};
            LO <= {LO[N-2:0], 1'b0};
            CNT = CNT-1;
     end
else if (SETL0)begin
            LO[0] <= 1;
            HI <= Diff;
       end

function integer log2 (input integer n);
       integer i;
       begin
      log2 = 1;
      for (i=0; 2**i<n; i=i+1)
            log2 = i+1;
        end
endfunction

endmodule;
```

```
module Divider_CU (output reg INIT, SHL, SETL0, DONE, input Start,
Sign, CNTEQ0, Reset, CLK);

parameter S0 = 3'b000, S1=3'b001, S2=3'b010, S3=3'b011,  S4=3'b100;

reg [2:0] state, next_state;

always @(posedge CLK, posedge Reset)
   if (Reset) state <= S0;
   else state <= next_state;

always @(state, Start, Sign, CNTEQ0) begin
   INIT=0; SHL=0; SETL0=0; DONE=0;
   case (state)
      S0:
        if (Start)  begin
           next_state=S1; INIT=1;  end
        else next_state=S0;
       S1: begin SHL=1; next_state=S2; end
       S2: begin
         if (!Sign)  SETL0=1;
          next_state=S3;
        end
       S3:
        if (CNTEQ0)  next_state=S4;
        else begin
           SHL=1; next_state=S2;   end
       S4: begin DONE=1;
        if (Start)  begin
           next_state=S1; INIT=1;  end
        else next_state=S4;
        end
       default: next_state='bx;
   endcase
end

endmodule
```

The ASMD chart for the circuit to compute the Average, Maximum and Minimum is given below:




The Verilog modules for the circuit to compute the Average, Maximum and Minimum are given below:

```
module AVG (output reg [3:0] AVG, output [3:0] MAX, MIN, output
DONE,input [3:0] DATA, input START, RESET, CLK);


wire [7:0] SUM;
wire [3:0] Q, R;
```

```
reg [7:0] B;

AVG_DP  M1 (SUM, MAX, MIN, CNTE0, DATA, INIT, UPDATE, CLK);

AVG_CU M2 (INIT, UPDATE, STARTD, DONE, START, DONED, CNTE0, RESET,
CLK);

Divider #(8) M3 (Q, R, DONED, SUM, B, STARTD, RESET, CLK);


// storing the number of scores in register B
always @ (posedge CLK)
   if (START)
      B <= DATA;


// computing the average
always @(DONED)
   if (2*R >= B)
     AVG = Q + 1;
   else
     AVG = Q;


endmodule


module AVG_DP (output reg [7:0] SUM, output reg [3:0] MAX, MIN, output
CNTE0, input [3:0] DATA, input INIT, UPDATE, CLK);

reg [3:0] CNT;

assign CNTE0 = ~| CNT;

always @(posedge CLK)

if (INIT) begin
     CNT <= DATA;
     SUM <= 0;
     MAX <= 0;
     MIN <= 4'b1111;
end
else if (UPDATE)begin
     SUM <= SUM + DATA;
     if (DATA > MAX) MAX <= DATA;
     if (DATA < MIN) MIN <= DATA;
     CNT <= CNT - 1;
end

endmodule
```

```
module AVG_CU  (output reg INIT, UPDATE, STARTD, DONE, input Start,
DONED, CNTE0, Reset, CLK);

parameter S0 = 2'b00, S1=2'b01, S2=2'b10, S3=2'b11;

reg [1:0] state, next_state;

always @(posedge CLK, posedge Reset)
   if (Reset) state <= S0;
   else state <= next_state;

always @(state, Start, DONED, CNTE0) begin
   INIT=0; UPDATE=0; STARTD=0; DONE=0;
   case (state)
      S0:
        if (Start)  begin
           next_state=S1; INIT=1;  end
        else next_state=S0;
       S1:
        if (CNTE0) begin STARTD=1; next_state=S2; end
        else begin
           UPDATE=1; next_state=S1;   end
       S2:
        if (DONED) begin DONE=1; next_state=S3; end
        else next_state=S2;
       S3: begin DONE=1;
        if (Start)  begin
           next_state=S1; INIT=1;  end
        else next_state=S3;
        end
       default: next_state='bx;
   endcase
end

endmodule
```

The following test bench has been used to test the correct functionality of the average circuit:

```
module AVG_Test();



wire [3:0] AVG, MAX, MIN;
wire DONE;

reg [3:0] DATA;

reg START, RESET, CLK;
```

```
AVG M1 (AVG, MAX, MIN, DONE, DATA, START, RESET, CLK);

initial begin
CLK = 0;
forever
#50 CLK = ~ CLK;
end

initial begin
RESET=1;
#100 RESET=0; START=1; DATA=4;
#100 START=0; DATA=5;
#100 DATA=9;
#100 DATA=15;
#100 DATA=1;
end


endmodule
```

The simulation snapshot is given below which indicates correct functionality of the circuit: