CA5-Report

Digital Systems I

Spring 1402-03
Digital Systems I – ECE894

Amirmortaza Rezaee
810101429

Design:

a.

Figure 1 illustrates the Data Path of this divider:

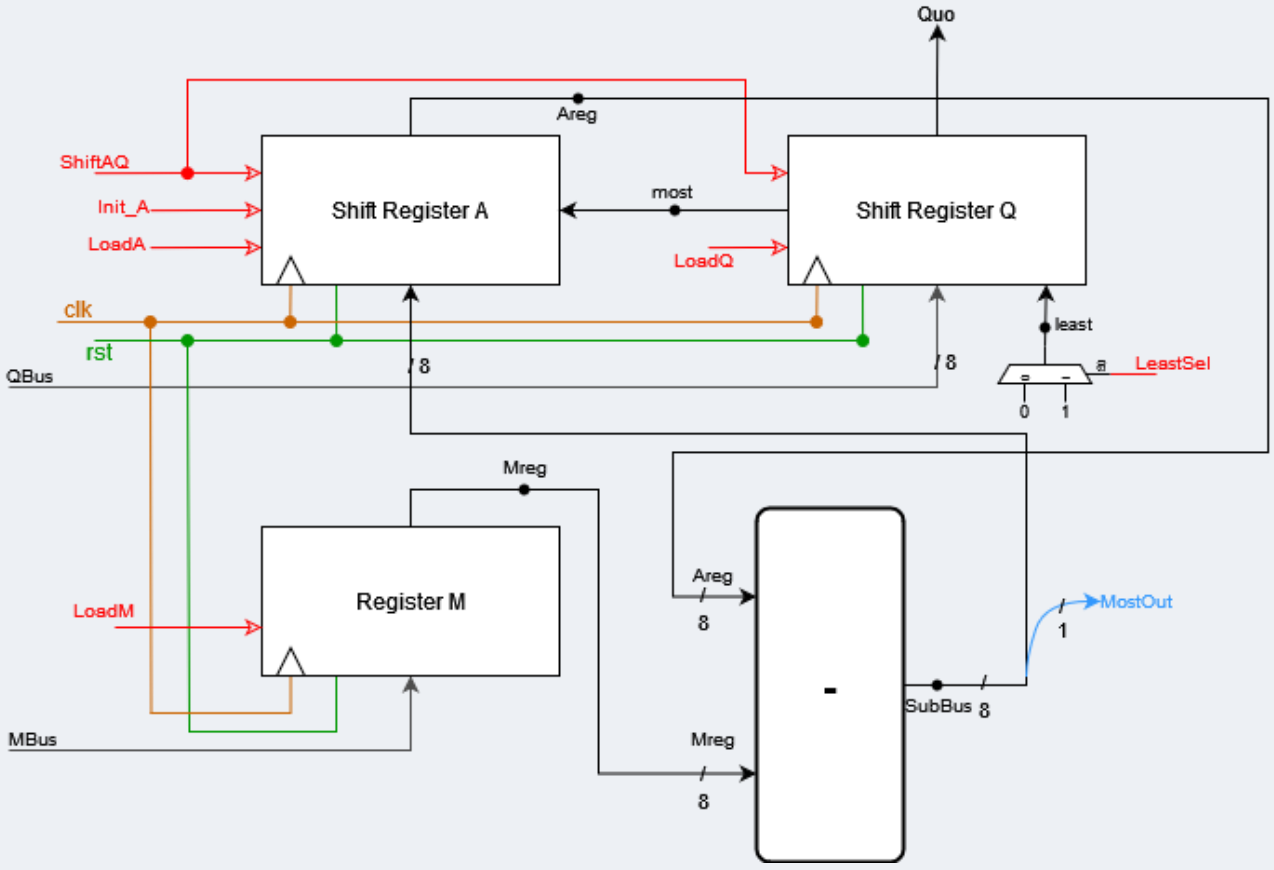


Figure 1

As shown above, this structure contains some input signals from the controlling unit which are shown in red, and a output signal (the one that is shown in blue.)

b.

This figure illustrates the Control Unit of the divider (Huffman Model):

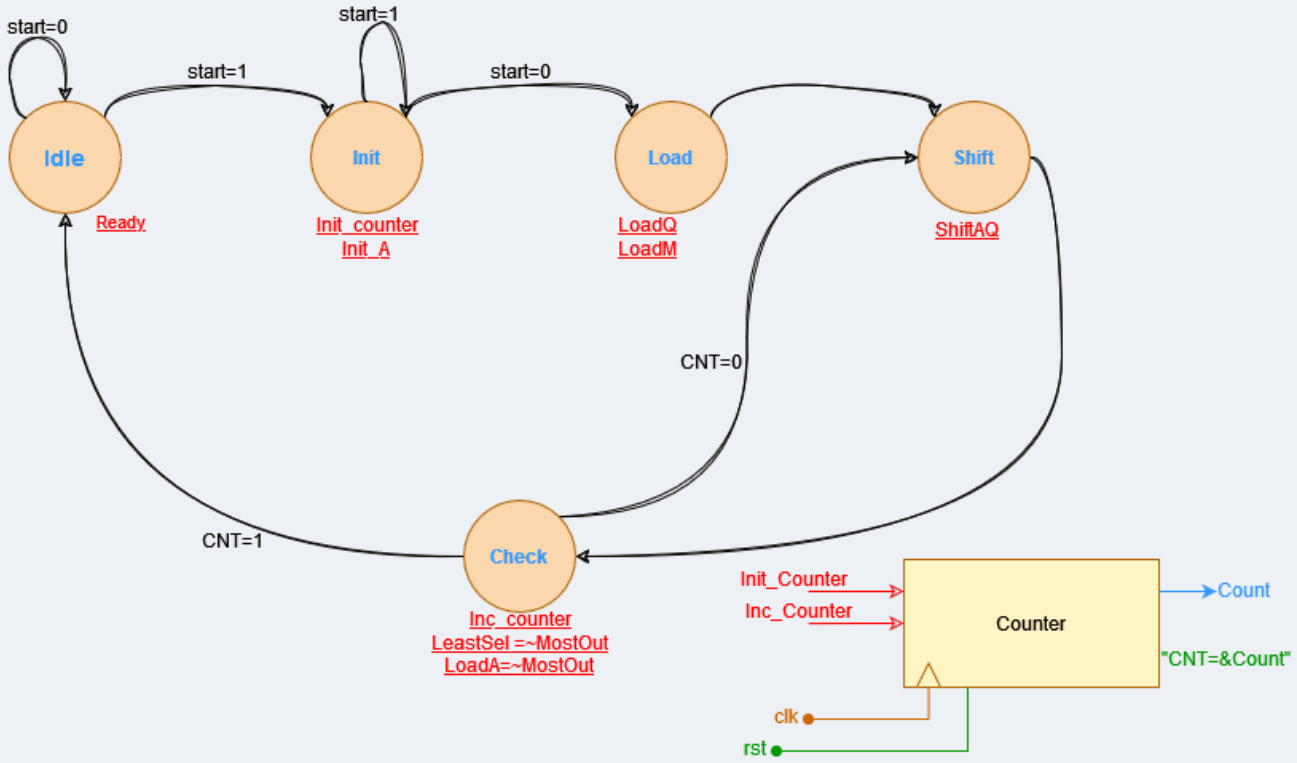


Figure 2

After the start signal occurred, we enter the initializing state in which the values of A and the counter initialized to zero. Then, after loading the values of Q and M, shifting starts and after that at the Check state, counter increments and if the MostOut signal was 0, LeastSel and LoadA become 1. So, the act of dividing with this restoring algorithm will be done properly.

Verilog:

The System Verilog description for the Data Path:

```
module DIVDP (input clk, rst, LoadM, LoadA, InitA, LoadQ, ShiftAQ, LeastSel,
              input[7:0] MBus, QBus, output MostOut, output[7:0] Rem, Quo);
    logic [7:0] Mreg, Areg, Qreg;
    wire [7:0] SubBus;
    wire least, most;

    always @(posedge clk, posedge rst) begin
        if (rst)
            Mreg <=8'b0;
        else
            if(LoadM)
                Mreg <= MBus;
    end

    always @(posedge clk, posedge rst) begin
        if (rst)
            Areg <= 8'b0;
        else begin
            if (InitA)
                Areg <= 8'b0;
            else begin
                if (ShiftAQ)
                    Areg <= {Areg[6:0],most};
                else if (LoadA)
                    Areg <=SubBus;
            end
        end
    end

    always @(posedge clk, posedge rst) begin
        if (rst)
            Qreg <=8'b0;
        else begin
            if (LoadQ)
                Qreg <= QBus;
            else if(ShiftAQ)
                Qreg <= {Qreg[6:0],1'b0};
            else if (LeastSel)
            Qreg[0] <= least;
        end
    end



    assign SubBus = Areg - Mreg;
    assign least = LeastSel;
    assign most = Qreg[7];
    assign MostOut = SubBus[7];
    assign Rem = Areg[7:0];
    assign Quo = Qreg[7:0];

endmodule
```

Figure 3

Figures 4 and 5 illustrate the Test Bench and simulation results for the Data Path:

```
module DIV_DP_TB ();
    logic clk = 1'b0;
    logic rst = 0;
    logic LoadM = 0 , LoadA = 0, InitA = 0, LoadQ = 0, ShiftAQ = 0, LeastSel = 0;
    logic [7:0] M;
    logic [7:0] Q;
    wire [7:0] Quotient, Reminder;
    wire MostOut;

    DIVDP UUT1(clk, rst, LoadM, LoadA, InitA, LoadQ, ShiftAQ, LeastSel,
               M, Q, MostOut, Reminder, Quotient);

    always #5 clk <= ~clk;

    initial begin
        #3 rst = 1;
        #3 rst = 0;
        #10 LoadM = 1'b1;
            M = 8'd7;
        #1 LoadM = 1'b0;
        #1 InitA = 1'b1;
        #1 LoadQ = 1'b1;
            Q = 8'd212;
        #1 LoadQ = 1'b0;
        #10 LeastSel = 1'b1;
        #7 LoadQ = 1'b1;
        #7 LoadQ = 1'b0;
        #10 LoadA = 1'b1;
        #10 ShiftAQ = 1'b1;
        #10 LeastSel = 1'b0;
        #300 $stop;
    end

endmodule
```

Figure 4

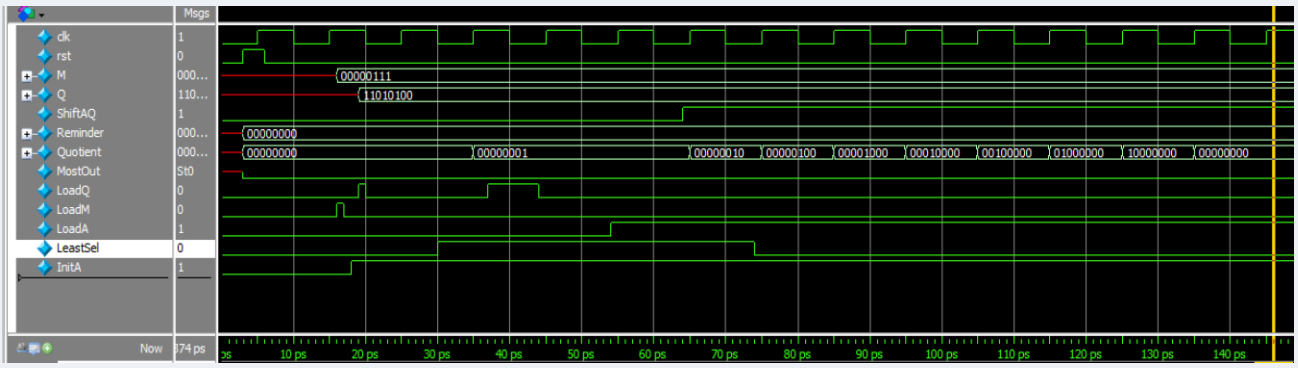


Figure 5

The System Verilog description for the Controller (Huffman Model):

```
module DIVCU (input clk, rst, start, MostOut,
              output logic LoadM, LoadA, InitA, LoadQ, ShiftAQ, LeastSel, Ready);
    wire CNT;
    logic Init_counter, Inc_counter;
    logic[2:0] pstate, nstate;
    logic[2:0] Count;
    parameter [2:0]
    Idle = 0, Init = 1 , Load = 2, Shift = 3, Check = 4;

    always @(pstate, start, CNT, MostOut) begin
        nstate = 0;
        {LoadM, LoadA, InitA, LoadQ, ShiftAQ, LeastSel, Ready} = 7'b0;
        {Init_counter, Inc_counter} = 2'b0;
            case(pstate)
                Idle: begin nstate = start ? Init : Idle; Ready = 1'b1; end
                Init: begin nstate = start ? Init : Load; {Init_counter, InitA} = 2'b11; end
                Load: begin nstate = Shift; {LoadM, LoadQ}=2'b11; end
                Shift: begin nstate = Check; ShiftAQ = 1'b1;  end
                Check: begin nstate = CNT ? Idle : Shift; Inc_counter=1'b1;
                             LeastSel=~MostOut; LoadA = ~MostOut; end
                default: nstate = Idle;
            endcase
    end

    always @(posedge clk, posedge rst) begin
        if (rst)
            pstate <= Idle;
        else
            pstate <= nstate;
    end

    always @(posedge clk, posedge rst) begin
        if (rst)
            Count <= 3'b0;
        else
            if (Init_counter) Count <= 3'd0;
            else
            if (Inc_counter) Count <= Count + 1;
    end

    assign CNT = &Count;

endmodule
```

Figure 6

Figures 7 and 8 illustrate the Test Bench and simulation results for the Controller Unit:

```
module DIV_CU_TB ();
    logic clk = 1'b0;
    logic rst = 0;
    logic start = 0, MostOut = 0;
    wire LoadM, LoadA, InitA, LoadQ, ShiftAQ, LeastSel, Ready;

    DIVCU UUT2(clk, rst, start, MostOut, LoadM, LoadA,
               InitA, LoadQ, ShiftAQ, LeastSel, Ready);

    always #5 clk <= ~clk;

    initial begin
        #3 rst = 1;
        #3 rst = 0;
        #10 start = 1'b1;
        #7 start = 1'b0;
        #10 MostOut = 1'b0;
        #30 MostOut = 1'b1;
        #300 $stop;
    end

endmodule
```

Figure 7

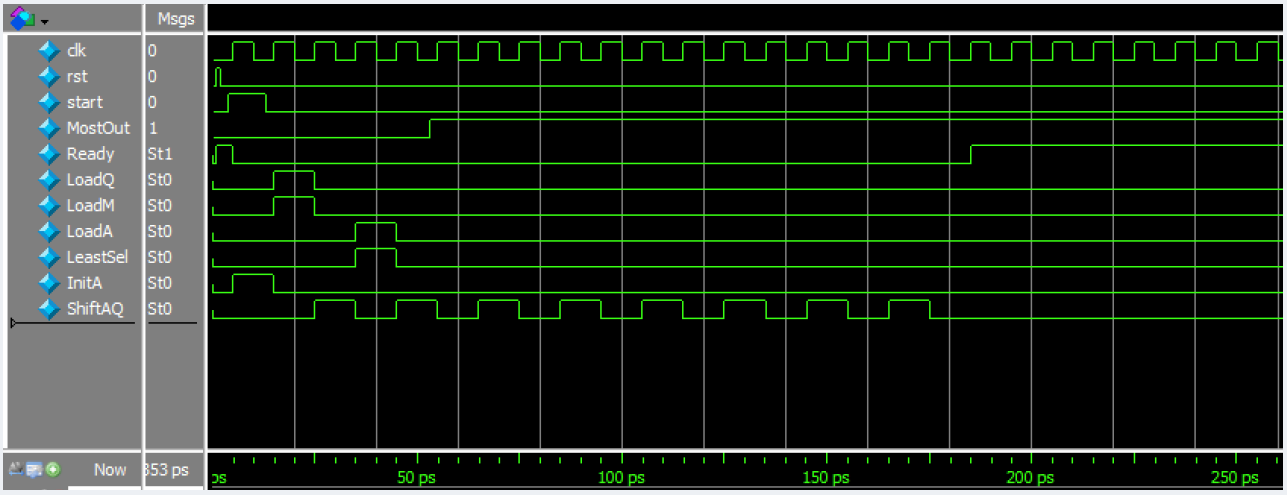


Figure 8

Simulation:

The Test Bench for this divider and the simulation results for several inputs are shown below:

```
module DIV_TOP (input clk, rst, start, input[7:0] M, Q, output[7:0] Quotient, Reminder, output Ready);
    wire LoadM, LoadA, InitA, LoadQ, ShiftAQ, MostOut, LeastSel, ASel;

    DIVDP dp(clk, rst, LoadM, LoadA, InitA, LoadQ, ShiftAQ, LeastSel, M, Q, MostOut, Reminder, Quotient);
    DIVCU cu(clk, rst, start, MostOut, LoadM, LoadA, InitA, LoadQ, ShiftAQ, LeastSel, Ready);

endmodule
```

Figure 9

```
`timescale 1ns/1ns
module DIV_TB ();
    logic clk = 1'b0;
    logic rst = 0;
    logic start = 0;
    logic [7:0] M;
    logic [7:0] Q;
    wire [7:0] Quotient, Reminder;
    wire Ready;

    DIV_TOP CUT1(clk, rst, start, M, Q, Quotient, Reminder, Ready);

    always #5 clk <= ~clk;

    initial begin
        #3 rst = 1;
        #3 rst = 0;
        #13 Q = 8'd171;
        #13 M = 8'd12;
        #3 start = 1;
        #13 start = 0;
        #300 $stop;
    end

endmodule
```

Figure 10

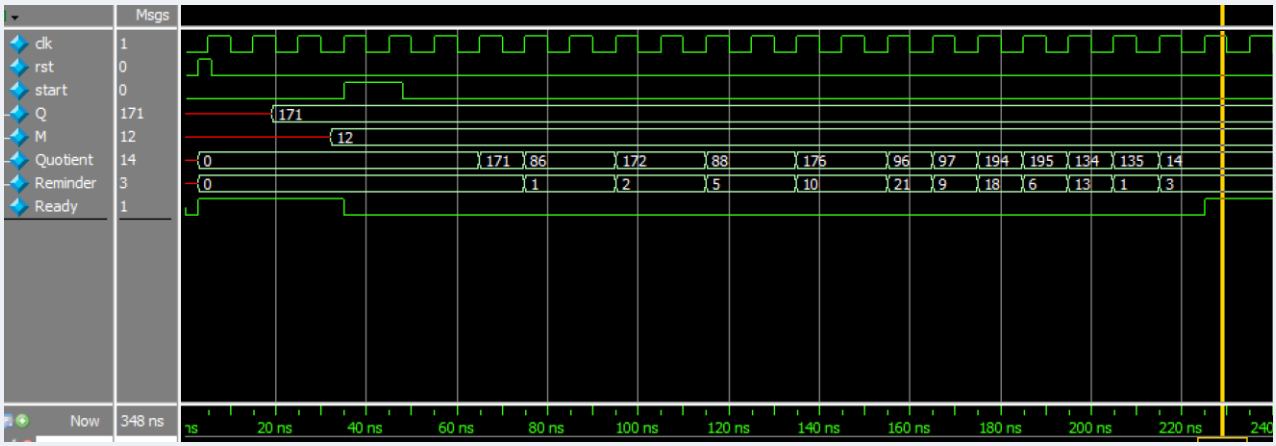


Figure 10 171÷12 → Q = 14, R = 3

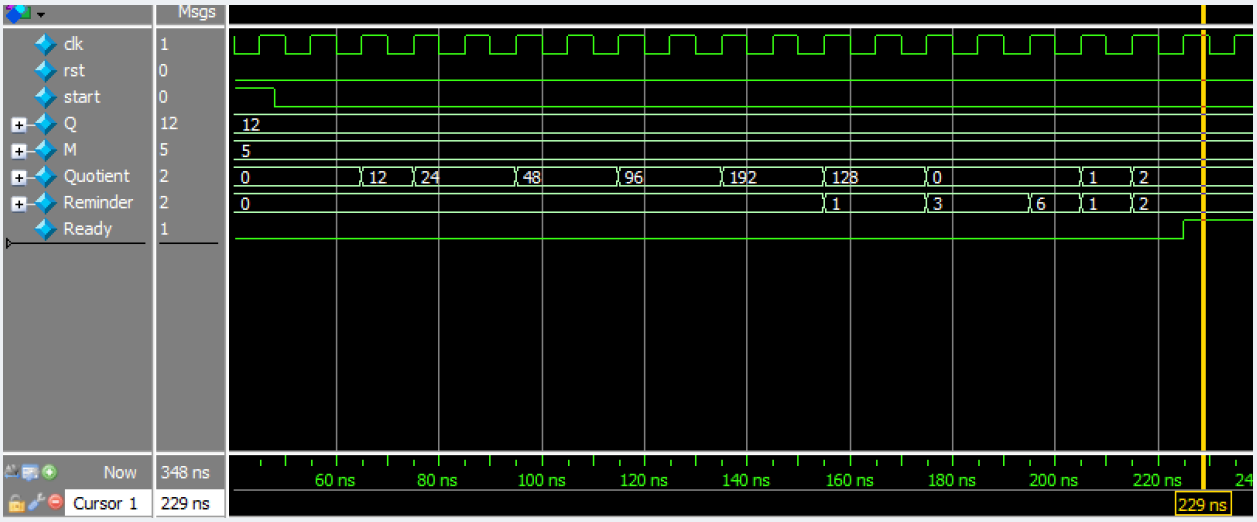


Figure 11 12÷5→ Q = 2, R = 2

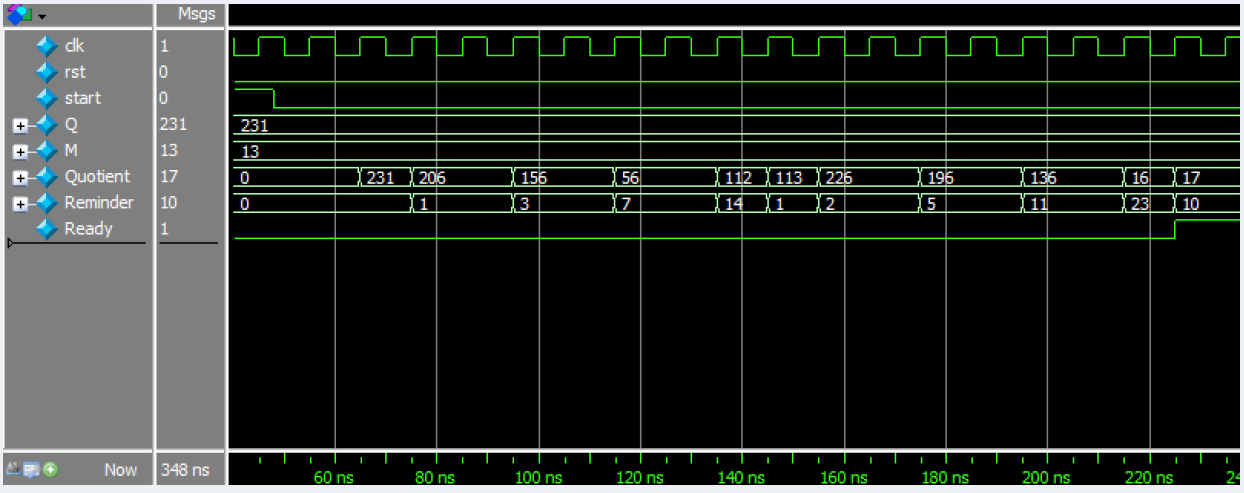


Figure 12 231÷13 → Q = 17, R = 10

Synthesize:

After synthesizing with Quartus and exporting .vo and .sdo files as outputs, a Test Bench has been written to test the outputs as described in videos:

```
`timescale 1ns/1ns
module DIV_pp_TB ();
    reg clk = 1'b0;
    reg rst = 0;
    reg start = 0;
    reg [7:0] M;
    reg [7:0] Q;
    wire [7:0] Quotient, Reminder;
    wire Ready;

    CA5TOPPP CUT2(clk, rst, start, M, Q, Quotient, Reminder, Ready);

    always #5 clk <= ~clk;

    initial begin
        #3 rst = 1;
        #3 rst = 0;
        #13 Q = 8'd171;
        #13 M = 8'd12;
        #3 start = 1;
        #13 start = 0;
        #300 $stop;
    end

endmodule
```

Figure 13

Here are some of the simulation results for the same cases tested before:

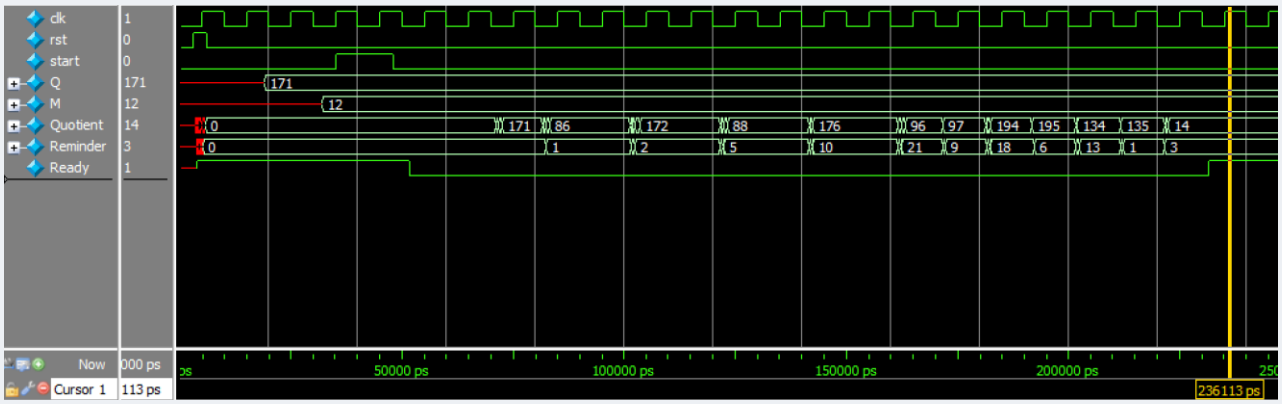


Figure 14 171÷12 → Q = 14, R = 3

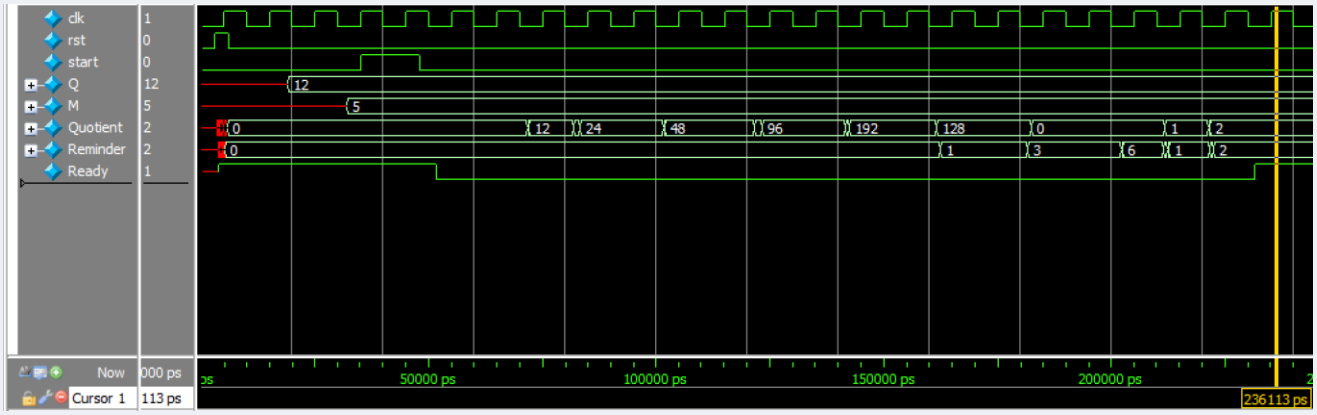


Figure 15 12÷5→ Q = 2, R = 2

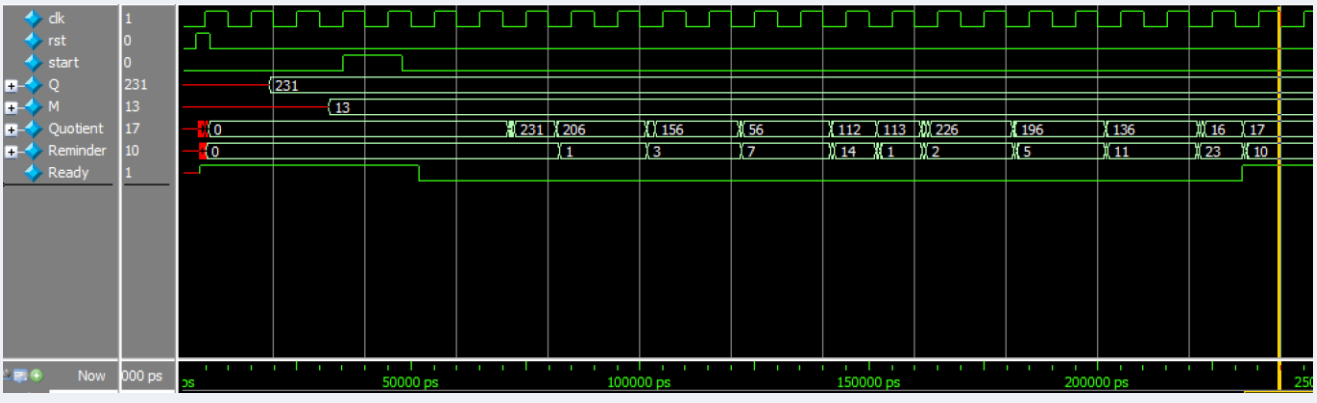
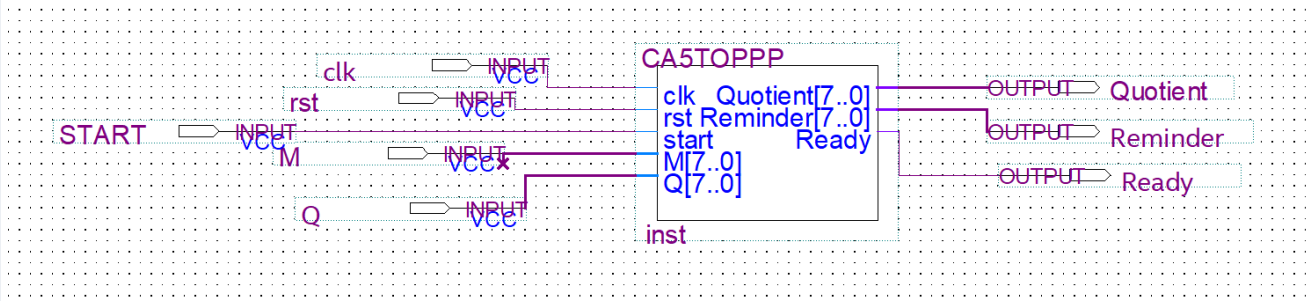


Figure 16 231÷13 → Q = 17, R = 10

As it's obvious, the synthesized circuit works properly.

Schematic:


CA5TOPPP
clk
rst
START
M
Q
INPUT
VCC
clk Quotient[7..0]
rst Reminder[7..0]
start Ready
M[7..0]
Q[7..0]
inst
OUTPUT
Quotient
Reminder
Ready