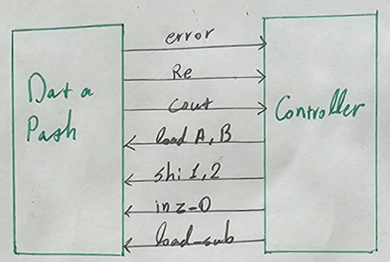
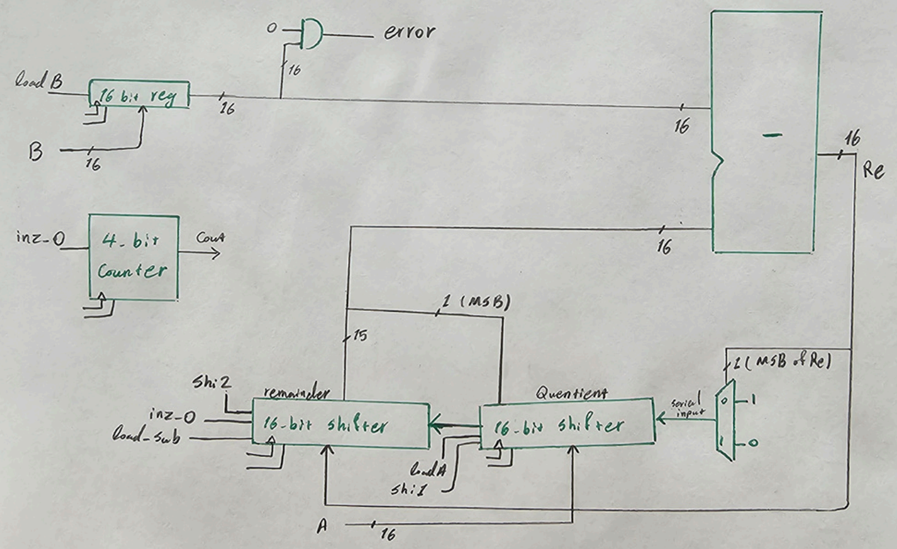
PROJECT REPORT

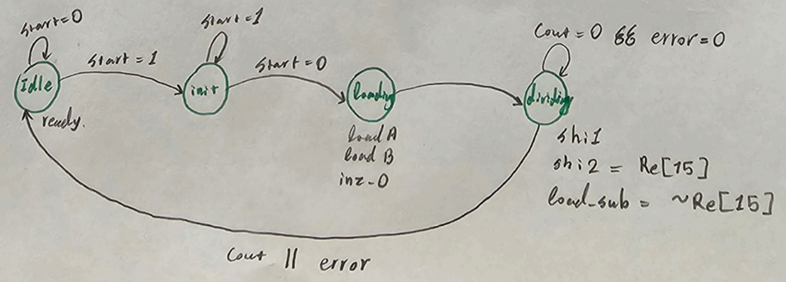
Parsa KafshduziBukani

CA5

Parsa Kafshdouzi Bahani 810102501 CA5







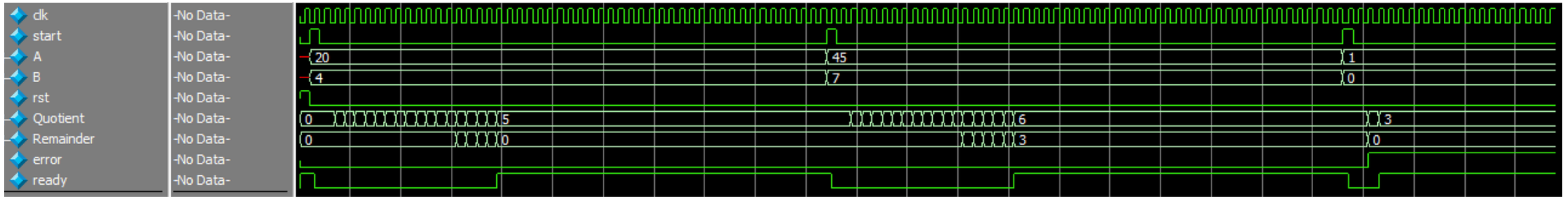
```
module Sequential_Divider (
    input [15:0] A,
    input [15:0] B,
    input load_A,
    input load_B,
    input sh1,
    input sh2,
    input inz_0,
    input load_sub,
    input rst, clk,
    output [15:0] Quotient,
    output [15:0] Remainder,
    output [15:0] Re,
    output cout,
    output reg error
);

reg [3:0] count;
reg [15:0] B_reg;
reg [15:0] Quotient_reg;
reg [15:0] Remainder_reg;
wire signed [15:0] subtracted;
wire serial_input;

always @(posedge clk or posedge rst) begin
    if (rst) begin
        B_reg <= 16'd0;
        error <= 0;
    end
    else if (load_B) begin
        B_reg <= B;
        if (B == 16'd0)
            error <= 1;
        else
            error <= 0;
    end
end

always @(posedge clk or posedge rst) begin
    if (rst)
        Quotient_reg <= 16'd0;
    else if (load_A)
        Quotient_reg <= A;
    else if (sh1)
        Quotient_reg <= {Quotient_reg[14:0], serial_input};
end

always @(posedge clk or posedge rst) begin
    if (rst)
        Remainder_reg <= 16'd0;
    else if (inz_0)
        Remainder_reg <= 16'd0;
    else if (load_sub)
        Remainder_reg <= subtracted;
    else if (sh2)
        Remainder_reg <= {Remainder_reg[14:0], Quotient_reg[15]};
end

always @(posedge clk or posedge rst) begin
    if (rst)
        count <= 4'd0;
    else if (inz_0)
        count <= 4'd0;
    else
        count <= count + 1;
end

assign cout = &count;
assign subtracted = {Remainder_reg[14:0], Quotient_reg[15]} - B_reg;
assign serial_input = subtracted[15] ? 1'b0 : 1'b1;

assign Quotient = Quotient_reg;
assign Remainder = Remainder_reg;
assign Re = subtracted;

endmodule
```

```
81  module Controller (
82      input clk, rst, start,
83      input [15:0] Re,
84      input cout,
85      input error,
86      output reg load_A, load_B,
87      output reg sh1, sh2,
88      output reg inz_0,
89      output reg load_sub,
90      output reg ready
91  );
92
93  parameter [1:0] idle = 2'b00, init = 2'b01, load = 2'b11, dividing = 2'b10;
94  reg [1:0] pstate, nstate;
95
96  always @(pstate, start, Re, cout, error) begin
97      nstate = 2'b0;
98      {ready, load_A, load_B, sh1, sh2, inz_0, load_sub} = 7'b0000000;
99
100     case(pstate)
101         idle: begin
102             ready = 1;
103             nstate = start ? init : idle;
104         end
105         init: begin
106             nstate = start ? init : load;
107         end
108         load: begin
109             nstate = dividing;
110             {load_A, load_B, inz_0} = 3'b111;
111         end
112         dividing: begin
113             nstate = cout ? idle : dividing;
114             sh1 = 1;
115             sh2 = Re[15] ? 1 : 0;
116             load_sub = Re[15] ? 0 : 1;
117             if (error)
118                 nstate = idle;
119         end
120     endcase
121 end
122
123 always @(posedge clk or posedge rst) begin
124     if (rst)
125         pstate <= idle;
126     else
127         pstate <= nstate;
128 end
129
130 endmodule
```

```
module Top_Level (
    input clk,
    input rst,
    input start,
    input [15:0] A,
    input [15:0] B,
    output [15:0] Quotient,
    output [15:0] Remainder,
    output ready,
    output error
);

wire load_A, load_B, sh1, sh2, inz_0, load_sub;
wire cout;
wire [15:0] Re;

Controller controller (
    .clk(clk),
    .rst(rst),
    .start(start),
    .Re(Re),
    .cout(cout),
    .error(error),
    .load_A(load_A),
    .load_B(load_B),
    .sh1(sh1),
    .sh2(sh2),
    .inz_0(inz_0),
    .load_sub(load_sub),
    .ready(ready)
);

Sequential_Divider sequential_divider (
    .A(A),
    .B(B),
    .load_A(load_A),
    .load_B(load_B),
    .sh1(sh1),
    .sh2(sh2),
    .inz_0(inz_0),
    .load_sub(load_sub),
    .rst(rst),
    .clk(clk),
    .Quotient(Quotient),
    .Remainder(Remainder),
    .Re(Re),
    .cout(cout),
    .error(error)
);

endmodule
```

```
3   module tb_Controller;
4
5       reg clk;
6       reg rst;
7       reg start;
8       reg [15:0] A;
9       reg [15:0] B;
10      wire [15:0] Quotient;
11      wire [15:0] Remainder;
12      wire error;
13      wire ready;
14
15      Top_Level uut (
16          .clk(clk),
17          .rst(rst),
18          .start(start),
19          .A(A),
20          .B(B),
21          .Quotient(Quotient),
22          .Remainder(Remainder),
23          .ready(ready),
24          .error(error)
25      );
26
27      initial begin
28          clk = 0;
29          forever #5 clk = ~clk;
30      end
31
```

```
32      initial begin
33          rst = 1;
34          start = 0;
35
36          #10;
37          rst = 0;
38
39          A = 16'd20;
40          B = 16'd4;
41          start = 1;
42          #10;
43          start = 0;
44
45          #500;
46
47          A = 16'd45;
48          B = 16'd7;
49          start = 1;
50          #10;
51          start = 0;
52
53          #500;
54
55          A = 16'd1;
56          B = 16'd0;
57          start = 1;
58          #10;
59          start = 0;
60
61          # 200 $stop;
62      end
63
64  endmodule
```


clk
start
A
B
rst
Quotient
Remainder
error
ready
-No Data-
20
45
1
4
7
0
0
5
6
3
0
0
3
0