```verilog
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: 
// Engineer: 
// 
// Create Date: 12/03/2020 10:25:40 PM
// Design Name: 
// Module Name: Divider_ControlLogic
// Project Name: 
// Target Devices: 
// Tool Versions: 
// Description: 
// 
// Dependencies: 
// 
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
// 
//////////////////////////////////////////////////////////////////////////////////


module Divider_ControlLogic(
    input EQ, LT, CEQ, CLT, Start, Ccase,
    input [4:0] PS,
    output [4:0] NS,
    output LD, Shift, ShiftR, Add, Sub0, SubB , Q, LDR, Next
//    input [3:0] EQ,
//    input [3:0] LT,
//    input Start,
//    input [6:0] PS,
//    output [6:0] NS,
//    output LD, Shift, Q, Sub, Add, NPrime, Prime
    );
    
    wire IDLE, SHIFT, SUBB, SUB0, OUT;
    wire Next_IDLE, Next_SHIFT, Next_SUBB, Next_SUB0, Next_OUT;
    
    // Present State
    assign IDLE     = PS[0];
    assign SHIFT    = PS[1];
    assign SUBB     = PS[2];
    assign SUB0     = PS[3];
    assign OUT      = PS[4];
    
    // Next State
    assign NS[0] = Next_IDLE;
    assign NS[1] = Next_SHIFT;
    assign NS[2] = Next_SUBB;
    assign NS[3] = Next_SUB0;
    assign NS[4] = Next_OUT;

    //Enter Logic
    assign Next_IDLE        = IDLE&~Start|OUT;
    assign Next_SHIFT       = IDLE&Start|SUBB&CLT|SUB0&CLT;
    assign Next_SUBB        = SHIFT&~LT|SHIFT&EQ;
    assign Next_SUB0        = SHIFT&LT;
    assign Next_OUT         = SUBB&CEQ|SUB0&CEQ;

    // Outputs
    assign LD       = IDLE;
    assign ShiftR   = SUBB&CLT|SUB0&CLT|IDLE&Start;
    //assign Shift    = SHIFT;
    assign Shift    = SHIFT&((~LT|EQ)|(LT))&~Ccase;
//    assign Shift    = SUBB|SUB0;
    assign LDR      = SHIFT&(~LT|EQ);
    assign Add      = SUBB|SUB0;
    assign Q        = SHIFT&(~LT|EQ);
    assign SubB     = SUBB;
    assign Sub0     = SUB0;
    assign Next     = OUT;


//    wire IDLE, DIVISOR, QUOTIENT, COUNT, CHECK;
//    wire Next_IDLE, Next_DIVISOR, Next_QUOTIENT, Next_COUNT, Next_CHECK;

//    // Present State
//    assign IDLE     = PS[0];
//    assign DIVISOR  = PS[1];
//    assign QUOTIENT = PS[2];
//    assign COUNT    = PS[3];
//    assign CHECK    = PS[4];

//    // Next State
//    assign NS[0] = Next_IDLE;
//    assign NS[1] = Next_DIVISOR;
//    assign NS[2] = Next_QUOTIENT;
//    assign NS[3] = Next_COUNT;
//    assign NS[4] = Next_CHECK;

//    //Enter Logic
//    assign Next_IDLE        = IDLE&~Start|DIVISOR&~EQ[0]&~LT[0]|CHECK&EQ[3];
//    assign Next_DIVISOR      = IDLE&Start|CHECK&~LT[3]&~EQ[3];
//    assign Next_QUOTIENT     = DIVISOR&(EQ[1]|LT[1])|COUNT&~EQ[2];
//    assign Next_COUNT        = QUOTIENT&((EQ[1]|~LT[1])|LT[1]);
//    assign Next_CHECK        = COUNT&EQ[2];

//    // Outputs
//    assign LD       = IDLE&Start;
//    assign Shift    = (DIVISOR&(EQ[0]|LT[0]))|(COUNT&~EQ[2]);
//    assign Q        = QUOTIENT&((~EQ[1]&~LT[1])|(EQ[1]|LT[1]));
//    assign SubB     = QUOTIENT&(~EQ[1]&~LT[1]);
//    assign Sub0     =
//    assign Add      = (CHECK&~EQ[3]&~LT[3]);
//    assign NPrime   = CHECK&EQ[3];
//    assign Prime    = DIVISOR&~EQ[0]&~LT[0];
endmodule
```