```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 01/31/2018 04:06:00 PM
// Design Name:
// Module Name: CountUD4L
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module CountUD4L(
    input clk,
    input up,
    input dw,
    input ld,
    input [3:0] Din,
    output [3:0] Q,
    output UTC,
    output DTC
    );

    wire [3:0] D, DUp, DDw, muxOut;

    // Arithmetic Logic
    assign DUp[0] = Q[0]^up;
    assign DUp[1] = Q[1]^(up&Q[0]) ;
    assign DUp[2] = Q[2]^(up&(&Q[1:0])) ;
    assign DUp[3] = Q[3]^(up&(&Q[2:0]));

    assign DDw[0] = Q[0]^dw;
    assign DDw[1] = Q[1]^(dw&~Q[0]);
    assign DDw[2] = Q[2]^(dw&(&(~Q[1:0])));
    assign DDw[3] = Q[3]^(dw&(&(~Q[2:0])));
```

```
    m4_1x4 ArithLogicMux (Q, DDw, DUp, DUp, {up, dw}, 1'b1, D);

    // Select count or load
    m2_1 mux1 (D[0], Din[0], ld, muxOut[0]);
    m2_1 mux2 (D[1], Din[1], ld, muxOut[1]);
    m2_1 mux3 (D[2], Din[2], ld, muxOut[2]);
    m2_1 mux4 (D[3], Din[3], ld, muxOut[3]);

    // flip flops
    FDRE #(.INIT(1'b0) ) ff1 (.C(clk), .R(1'b0), .CE(1'b1), .D(muxOut[0]), .Q(Q[0]));
    FDRE #(.INIT(1'b0) ) ff2 (.C(clk), .R(1'b0), .CE(1'b1), .D(muxOut[1]), .Q(Q[1]));
    FDRE #(.INIT(1'b0) ) ff3 (.C(clk), .R(1'b0), .CE(1'b1), .D(muxOut[2]), .Q(Q[2]));
    FDRE #(.INIT(1'b0) ) ff4 (.C(clk), .R(1'b0), .CE(1'b1), .D(muxOut[3]), .Q(Q[3]));

    // generate terminal counts
    assign UTC = &Q;
    assign DTC = &(~Q);

endmodule
```