```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: 
// Engineer: 
// 
// Create Date: 10/17/2023 02:42:13 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,        //  the system clock,
    input UP,           //  (increment)
    input DW,           //  (decrement)
    input LD,           //  (load control)
    input [3:0] Din,          //  the 4-bit bus Din that will be loaded on the clock 
edge if LD is high.
    
    output [3:0] Q,           //  a 4-bit bus Q which is the current value held by 
the counter,
    output UTC,         //  the signal UTC (UP Terminal Count) which is 1 only when 
the counter is at 4'b1111 (15 in decimal), and
    output DTC          //  the signal DTC (Down Terminal Count) which is 1 only 
when the counter is at 4'b0000.
    );
    
    wire Inc;
    assign Inc = (UP ^ DW) | LD;
    wire UxDoL;
    assign UxDoL = (UP ^ DW | LD);  // Different logic, order of operations matters. 
The Dwire[s] use this order of operations
    
    wire Dwire [3:0];
    assign Dwire[0] = ((Q[0] ^ UxDoL)                         & UP & ~DW) & ~LD  |
(((Q[0] ^ (UxDoL)) | (LD & Din[0])            & ~UP & DW) & ~LD);
    assign Dwire[1] = ((Q[1] ^ (UxDoL & Q[0]))                & UP & ~DW  & ~LD) |
(((Q[1] ^ ((UxDoL) & ~Q[0]))                  & ~UP & DW) & ~LD) | (LD & Din[1]);
    assign Dwire[2] = ((Q[2] ^ (UxDoL & Q[0] & Q[1]))         & UP & ~DW) & ~LD  |
(((Q[2] ^ ((UxDoL) & ~Q[1] & ~Q[0]))          & ~UP & DW) & ~LD) | (LD & Din[2]);
    assign Dwire[3] = ((Q[3] ^ (UxDoL & Q[0] & Q[1] & Q[2]))  & UP & ~DW  & ~LD) |
(((Q[3] ^ ((UxDoL) & ~Q[1] & ~Q[2] & ~Q[0])) & ~UP & DW) & ~LD) | (LD & Din[3]);

    assign UTC = &(Q[3:0]);
    assign DTC = &(~Q[3:0]);

    FDRE #(.INIT(1'b0)) b0_ff (.C(clk), .R(1'b0), .CE(Inc), .D(Dwire[0]), .Q(Q[0]));
    //  From "Using Flip-Flops in Vivado"
https://classes.soe.ucsc.edu/cse100/Fall23/lab/FDRE/FDRE.html
    FDRE #(.INIT(1'b0)) b1_ff (.C(clk), .R(1'b0), .CE(Inc), .D(Dwire[1]), .Q(Q[1]));
    FDRE #(.INIT(1'b0)) b2_ff (.C(clk), .R(1'b0), .CE(Inc), .D(Dwire[2]), .Q(Q[2]));
    FDRE #(.INIT(1'b0)) b3_ff (.C(clk), .R(1'b0), .CE(Inc), .D(Dwire[3]), .Q(Q[3]));

endmodule
```