```
// A driver for the 16x16x2 LED display expansion board.
// Read below for an overview of the ports.
// IMPORTANT: You do not need to necessarily modify this file. But if you do, be sure you know what you are doing.

// FREQDIV: (Parameter) Sets the scanning speed (how often the display cycles through rows)
//          The CLK input divided by 2^(FREQDIV) is the interval at which the driver switches rows.
// GPIO_1: (Output) The 36-pin GPIO1 header, as on the DE1-SoC board.
// RedPixels: (Input) A 16x16 array of logic items corresponding to the red pixels you'd like to have lit on the display.
// GrnPixels: (Input) A 16x16 array of logic items corresponding to the green pixels you'd like to have lit on the display.
// EnableCount: (Input) Whether to continue moving through the rows.
// CLK: (Input) The system clock.
// RST: (Input) Resets the display driver. Required during startup before use.
module LEDDriver #(parameter FREQDIV = 0) (GPIO_1, RedPixels, GrnPixels, EnableCount, CLK, RST);
    output logic [35:0] GPIO_1;
    input logic [15:0][15:0] RedPixels ;
    input logic [15:0][15:0] GrnPixels ;
    input logic EnableCount, CLK, RST;

    reg [(FREQDIV + 3):0] Counter;
    logic [3:0] RowSelect;
    assign RowSelect = Counter[(FREQDIV + 3):FREQDIV];

    always_ff @(posedge CLK)
    begin
        if(RST) Counter <= 'b0;
        if(EnableCount) Counter <= Counter + 1'b1;
    end

    assign GPIO_1[35:32] = RowSelect;
    assign GPIO_1[31:16] = { GrnPixels[RowSelect][0], GrnPixels[RowSelect][1], GrnPixels[RowSelect][2], GrnPixels[RowSelect][3], GrnPixels[RowSelect][4], GrnPixels[RowSelect][5], GrnPixels[RowSelect][6], GrnPixels[RowSelect][7], GrnPixels[RowSelect][8], GrnPixels[RowSelect][9], GrnPixels[RowSelect][10], GrnPixels[RowSelect][11], GrnPixels[RowSelect][12], GrnPixels[RowSelect][13], GrnPixels[RowSelect][14], GrnPixels[RowSelect][15] };
    assign GPIO_1[15:0] = { RedPixels[RowSelect][0], RedPixels[RowSelect][1], RedPixels[RowSelect][2], RedPixels[RowSelect][3], RedPixels[RowSelect][4], RedPixels[RowSelect][5], RedPixels[RowSelect][6], RedPixels[RowSelect][7], RedPixels[RowSelect][8], RedPixels[RowSelect][9], RedPixels[RowSelect][10], RedPixels[RowSelect][11], RedPixels[RowSelect][12], RedPixels[RowSelect][13], RedPixels[RowSelect][14], RedPixels[RowSelect][15] };
endmodule

module LEDDriver_Test();
    logic CLK, RST, EnableCount;
    logic [15:0][15:0]RedPixels;
    logic [15:0][15:0]GrnPixels;
    logic [35:0] GPIO_1;

    LEDDriver #(.FREQDIV(2)) Driver(.GPIO_1, .RedPixels, .GrnPixels, .EnableCount, .CLK, .RST);

    initial
    begin
        CLK <= 1'b0;
        forever #50 CLK <= ~CLK;
    end

    initial
    begin
        EnableCount <= 1'b0;
        RedPixels <= '{default:0};
        GrnPixels <= '{default:0};
        @(posedge CLK);

        RST <= 1; @(posedge CLK);
        RST <= 0; @(posedge CLK);
        @(posedge CLK); @(posedge CLK); @(posedge CLK);

        GrnPixels[1][1] <= 1'b1; @(posedge CLK);
        EnableCount <= 1'b1; @(posedge CLK); #1000;
        RedPixels[2][2] <= 1'b1;
        RedPixels[2][3] <= 1'b1;
        GrnPixels[2][3] <= 1'b1; @(posedge CLK); #1000;
        EnableCount <= 1'b0; @(posedge CLK); #1000;
        GrnPixels[1][1] <= 1'b0; @(posedge CLK);
        $stop;

    end
endmodule

module LEDDriver_TestPhysical (CLOCK_50, RST, Speed, GPIO_1);
    input logic CLOCK_50, RST;
    input logic [9:0] Speed;
    output logic [35:0] GPIO_1;
    logic [15:0][15:0]RedPixels;
    logic [15:0][15:0]GrnPixels;
    logic [31:0] Counter;
    logic EnableCount;

    LEDDriver #(.FREQDIV(15)) Driver (.CLK(CLOCK_50), .RST, .EnableCount, .RedPixels, .
GrnPixels, .GPIO_1);

    //                       F E D C B A 9 8 7 6 5 4 3 2 1 0
    assign RedPixels[00] = '{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1};
    assign RedPixels[01] = '{1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1};
    assign RedPixels[02] = '{1,0,1,1,1,1,1,1,1,1,1,1,1,1,0,1};
    assign RedPixels[03] = '{1,0,1,1,0,0,0,0,0,0,0,0,1,1,0,1};
    assign RedPixels[04] = '{1,0,1,0,1,1,1,1,1,1,1,1,0,1,0,1};
    assign RedPixels[05] = '{1,0,1,0,1,1,0,0,0,0,1,1,0,1,0,1};
    assign RedPixels[06] = '{1,0,1,0,1,0,1,1,1,1,0,1,0,1,0,1};
    assign RedPixels[07] = '{1,0,1,0,1,0,1,0,1,1,0,1,0,1,0,1};
    assign RedPixels[08] = '{1,0,1,0,1,0,1,1,0,1,0,1,0,1,0,1};
    assign RedPixels[09] = '{1,0,1,0,1,0,1,1,1,1,0,1,0,1,0,1};
    assign RedPixels[10] = '{1,0,1,0,1,1,0,0,0,0,1,1,0,1,0,1};
    assign RedPixels[11] = '{1,0,1,0,1,1,1,1,1,1,1,1,0,1,0,1};
    assign RedPixels[12] = '{1,0,1,1,0,0,0,0,0,0,0,0,1,1,0,1};
    assign RedPixels[13] = '{1,0,1,1,1,1,1,1,1,1,1,1,1,1,0,1};
    assign RedPixels[14] = '{1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1};
    assign RedPixels[15] = '{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1};

    assign GrnPixels[00] = '{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1};
    assign GrnPixels[01] = '{0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0};
    assign GrnPixels[02] = '{0,1,1,0,0,0,0,0,0,0,0,0,0,1,1,0};
    assign GrnPixels[03] = '{0,1,0,1,1,1,1,1,1,1,1,1,1,0,1,0};
    assign GrnPixels[04] = '{0,1,0,1,1,0,0,0,0,0,0,1,1,0,1,0};
    assign GrnPixels[05] = '{0,1,0,1,0,1,1,1,1,1,1,0,1,0,1,0};
    assign GrnPixels[06] = '{0,1,0,1,0,1,1,0,0,1,1,0,1,0,1,0};
    assign GrnPixels[07] = '{0,1,0,1,0,1,0,1,0,0,1,0,1,0,1,0};
    assign GrnPixels[08] = '{0,1,0,1,0,1,0,0,1,0,1,0,1,0,1,0};
    assign GrnPixels[09] = '{0,1,0,1,0,1,1,0,0,1,1,0,1,0,1,0};
    assign GrnPixels[10] = '{0,1,0,1,0,1,1,1,1,1,1,0,1,0,1,0};
    assign GrnPixels[11] = '{0,1,0,1,1,0,0,0,0,0,0,1,1,0,1,0};
    assign GrnPixels[12] = '{0,1,0,1,1,1,1,1,1,1,1,1,1,0,1,0};
    assign GrnPixels[13] = '{0,1,1,0,0,0,0,0,0,0,0,0,0,1,1,0};
    assign GrnPixels[14] = '{0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0};
    assign GrnPixels[15] = '{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1};

    always_ff @(posedge CLOCK_50)
    begin
        if(RST) Counter <= 'b0;
        else
        begin
            Counter <= Counter + 1'b1;
            if(Counter >= Speed)
            begin
                EnableCount <= 1'b1;
                Counter <= 'b0;
            end
            else EnableCount <= 1'b0;
        end
    end
endmodule
```