Controle de Periféricos - Projeto 2 - `led_twinkle`

Controle de Periféricos - Projeto 2 - `led_twinkle`

Projeto em funcionamento

Código do projeto

Módulo `led_twinkle`

```
module led_twinkle( input wire clk, output wire [3:0] leds );
        // Contador usado como variavel intermediaria
        reg [25:0] counter = 26'd0;

        // Iterador que ira indicar quais LEDs deverao ser acendidos
        reg [3:0] iter = 4'd0;

        // Estado dos LEDs
        reg [3:0] leds_r = 4'b1111;

        // Sinal para resetar o contador
        reg reset_cnt = 1'd0;

        // Bloco always executado nas bordas de subida do clock
        always @ ( posedge clk )
        begin
            // Incrementar 'counter'
            counter <= counter + 26'd1;

            // Se o valor de counter for 50.000.000
            if( counter == 26'd50000000 ) begin
                // Alterar o estado de 'reset_cnt'
                reset_cnt <= ~reset_cnt;

                // Reiniciar 'counter'
                counter <= 26'd0;
            end
        end

        always @ ( reset_cnt )
        begin
            // Incrementar 'iter'
            if( iter < 4'd7 ) begin
                iter = iter + 4'd1;
            end
            else begin
                iter = 4'd0;
```

```
            end

            // Acender os LEDs de acordo com 'iter'
            case( iter )
                4'd1: leds_r=4'b1110;
                4'd2: leds_r=4'b1100;
                4'd3: leds_r=4'b1000;
                4'd4: leds_r=4'b0000;
                4'd5: leds_r=4'b0001;
                4'd6: leds_r=4'b0011;
                4'd7: leds_r=4'b0111;
                default: leds_r=4'b1111;
            endcase
        end

        // --> Atribuicao aos LEDs
        assign leds = leds_r;
endmodule
```

Módulo principal

```
module demo_ep4ce6( input wire FPGA_CLK, output wire [3:0] LED );
    led_twinkle piscar ( .clk(FPGA_CLK), .leds(LED) );
endmodule
```