Persistence of Vision LED display

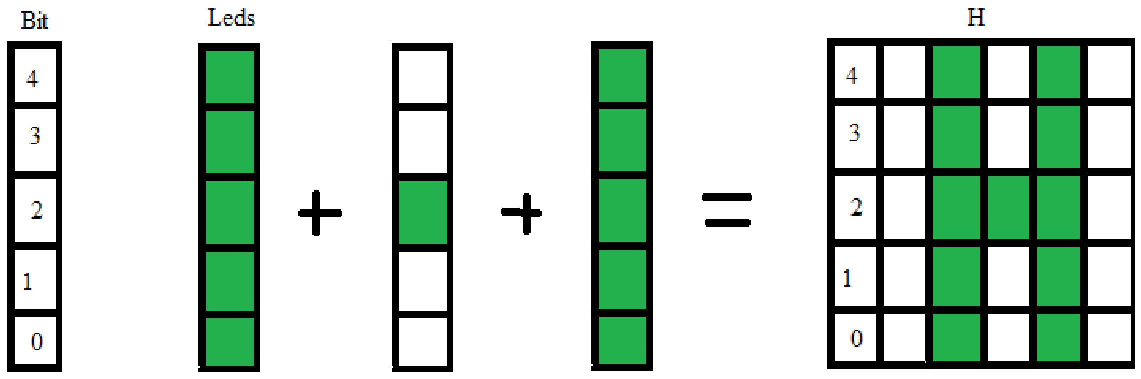
ELEX 7660

Russell Nicolas

Introduction

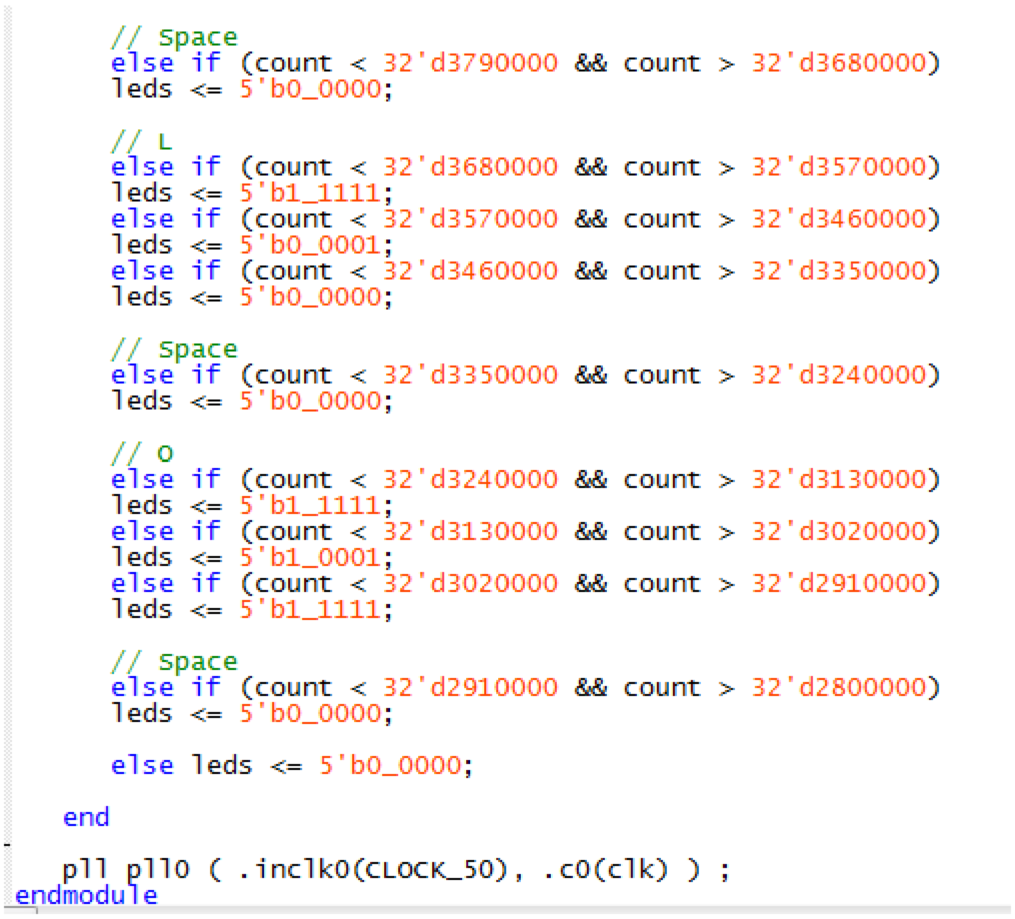
Persistence of vision is a type of optical illusion that takes advantage of "still frames" in time. In real life there are no "still frames" but the human eye can only perceive time at a certain rate. Using this optical illusion, flashing LEDs at a certain rate as it passes through time can emulate a still image.

Background



Working Code

```
module project2 ( input logic CLOCK_50,         // 50 MHz clock
                  input logic mag,              // Magnet sensor
                  output logic [4:0] leds);     // 5 LEDs

   logic [31:0] count ;


   always_ff@(posedge CLOCK_50) begin
   if (!mag)
      count <= 0;      // turn off LEDs when passes by magnet
   else                // reset counter
      count <= count ? count - 1'd1 : 32'd5000000;
   end


   // Spell "HELLO" with LEDs
   always_ff@(posedge CLOCK_50) begin

      // H
      if (count < 32'd5000000 && count > 32'd4890000)
      leds <= 5'b1_1111;
      else if (count < 32'd4890000 && count > 32'd4780000)
      leds <= 5'b0_0100;
      else if (count < 32'd4780000 && count > 32'd4670000)
      leds <= 5'b1_1111;

      // Space
      else if (count < 32'd4670000 && count > 32'd4560000)
      leds <= 5'b0_0000;

      // E
      else if (count < 32'd4560000 && count > 32'd4450000)
      leds <= 5'b1_1111;
      else if (count < 32'd4450000 && count > 32'd4340000)
      leds <= 5'b1_0101;
      else if (count < 32'd4340000 && count > 32'd4230000)
      leds <= 5'b1_0001;

      // Space
      else if (count < 32'd4230000 && count > 32'd4120000)
      leds <= 5'b0_0000;

      // L
      else if (count < 32'd4120000 && count > 32'd4010000)
      leds <= 5'b1_1111;
      else if (count < 32'd4010000 && count > 32'd3900000)
      leds <= 5'b0_0001;
      else if (count < 32'd3900000 && count > 32'd3790000)
      leds <= 5'b0_0000;
```



```
      // Space
      else if (count < 32'd3790000 && count > 32'd3680000)
      leds <= 5'b0_0000;

      // L
      else if (count < 32'd3680000 && count > 32'd3570000)
      leds <= 5'b1_1111;
      else if (count < 32'd3570000 && count > 32'd3460000)
      leds <= 5'b0_0001;
      else if (count < 32'd3460000 && count > 32'd3350000)
      leds <= 5'b0_0000;

      // Space
      else if (count < 32'd3350000 && count > 32'd3240000)
      leds <= 5'b0_0000;

      // O
      else if (count < 32'd3240000 && count > 32'd3130000)
      leds <= 5'b1_1111;
      else if (count < 32'd3130000 && count > 32'd3020000)
      leds <= 5'b1_0001;
      else if (count < 32'd3020000 && count > 32'd2910000)
      leds <= 5'b1_1111;

      // Space
      else if (count < 32'd2910000 && count > 32'd2800000)
      leds <= 5'b0_0000;

      else leds <= 5'b0_0000;

   end

   pll pll0 ( .inclk0(CLOCK_50), .c0(clk) ) ;
endmodule
```

```
module pll ( inclk0, c0);

        input     inclk0;
        output    c0;

        wire [0:0] sub_wire2 = 1'h0;
        wire [4:0] sub_wire3;
        wire  sub_wire0 = inclk0;
        wire [1:0] sub_wire1 = {sub_wire2, sub_wire0};
        wire [0:0] sub_wire4 = sub_wire3[0:0];
        wire  c0 = sub_wire4;

        altpll altpll_component ( .inclk (sub_wire1), .clk
          (sub_wire3), .activeclock (), .areset (1'b0), .clkbad
          (), .clkena ({6{1'b1}}), .clkloss (), .clkswitch
          (1'b0), .configupdate (1'b0), .enable0 (), .enable1 (),
          .extclk (), .extclkena ({4{1'b1}}), .fbin (1'b1),
          .fbmimicbidir (), .fbout (), .fref (), .icdrclk (),
          .locked (), .pfdena (1'b1), .phasecounterselect
          ({4{1'b1}}), .phasedone (), .phasestep (1'b1),
          .phaseupdown (1'b1), .pllena (1'b1), .scanaclr (1'b0),
          .scanclk (1'b0), .scanclkena (1'b1), .scandata (1'b0),
          .scandataout (), .scandone (), .scanread (1'b0),
          .scanwrite (1'b0), .sclkout0 (), .sclkout1 (),
          .vcooverrange (), .vcounderrange ());

        defparam
                altpll_component.bandwidth_type = "AUTO",
                altpll_component.clk0_divide_by =  25000,
                altpll_component.clk0_duty_cycle = 50,
                altpll_component.clk0_multiply_by = 1,
                altpll_component.clk0_phase_shift = "0",
                altpll_component.compensate_clock = "CLK0",
                altpll_component.inclk0_input_frequency = 20000,
                altpll_component.intended_device_family = "Cyclone IV E",
                altpll_component.lpm_hint = "CBX_MODULE_PREFIX=lab1clk",
                altpll_component.lpm_type = "altpll",
                altpll_component.operation_mode = "NORMAL",
                altpll_component.pll_type = "AUTO",
                altpll_component.port_activeclock = "PORT_UNUSED",
                altpll_component.port_areset = "PORT_UNUSED",
                altpll_component.port_clkbad0 = "PORT_UNUSED",
                altpll_component.port_clkbad1 = "PORT_UNUSED",
                altpll_component.port_clkloss = "PORT_UNUSED",
                altpll_component.port_clkswitch = "PORT_UNUSED",
                altpll_component.port_configupdate = "PORT_UNUSED",
                altpll_component.port_fbin = "PORT_UNUSED",
                altpll_component.port_inclk0 = "PORT_USED",
                altpll_component.port_inclk1 = "PORT_UNUSED",
                altpll_component.port_locked = "PORT_UNUSED",
                altpll_component.port_pfdena = "PORT_UNUSED",
                altpll_component.port_phasecounterselect = "PORT_UNUSED",
                altpll_component.port_phasedone = "PORT_UNUSED",
                altpll_component.port_phasestep = "PORT_UNUSED",
                altpll_component.port_phaseupdown = "PORT_UNUSED",
                altpll_component.port_pllena = "PORT_UNUSED",
                altpll_component.port_scanaclr = "PORT_UNUSED",
                altpll_component.port_scanclk = "PORT_UNUSED",
                altpll_component.port_scanclkena = "PORT_UNUSED",
                altpll_component.port_scandata = "PORT_UNUSED",
                altpll_component.port_scandataout = "PORT_UNUSED",
                altpll_component.port_scandone = "PORT_UNUSED",
                altpll_component.port_scanread = "PORT_UNUSED",
                altpll_component.port_scanwrite = "PORT_UNUSED",
                altpll_component.port_clk0 = "PORT_USED",
                altpll_component.port_clk1 = "PORT_UNUSED",
                altpll_component.port_clk2 = "PORT_UNUSED",
                altpll_component.port_clk3 = "PORT_UNUSED",
                altpll_component.port_clk4 = "PORT_UNUSED",
                altpll_component.port_clk5 = "PORT_UNUSED",
                altpll_component.port_clkena0 = "PORT_UNUSED",
                altpll_component.port_clkena1 = "PORT_UNUSED",
                altpll_component.port_clkena2 = "PORT_UNUSED",
                altpll_component.port_clkena3 = "PORT_UNUSED",
                altpll_component.port_clkena4 = "PORT_UNUSED",
                altpll_component.port_clkena5 = "PORT_UNUSED",
                altpll_component.port_extclk0 = "PORT_UNUSED",
                altpll_component.port_extclk1 = "PORT_UNUSED",
                altpll_component.port_extclk2 = "PORT_UNUSED",
                altpll_component.port_extclk3 = "PORT_UNUSED",
                altpll_component.width_clock = 5;

endmodule
```

References