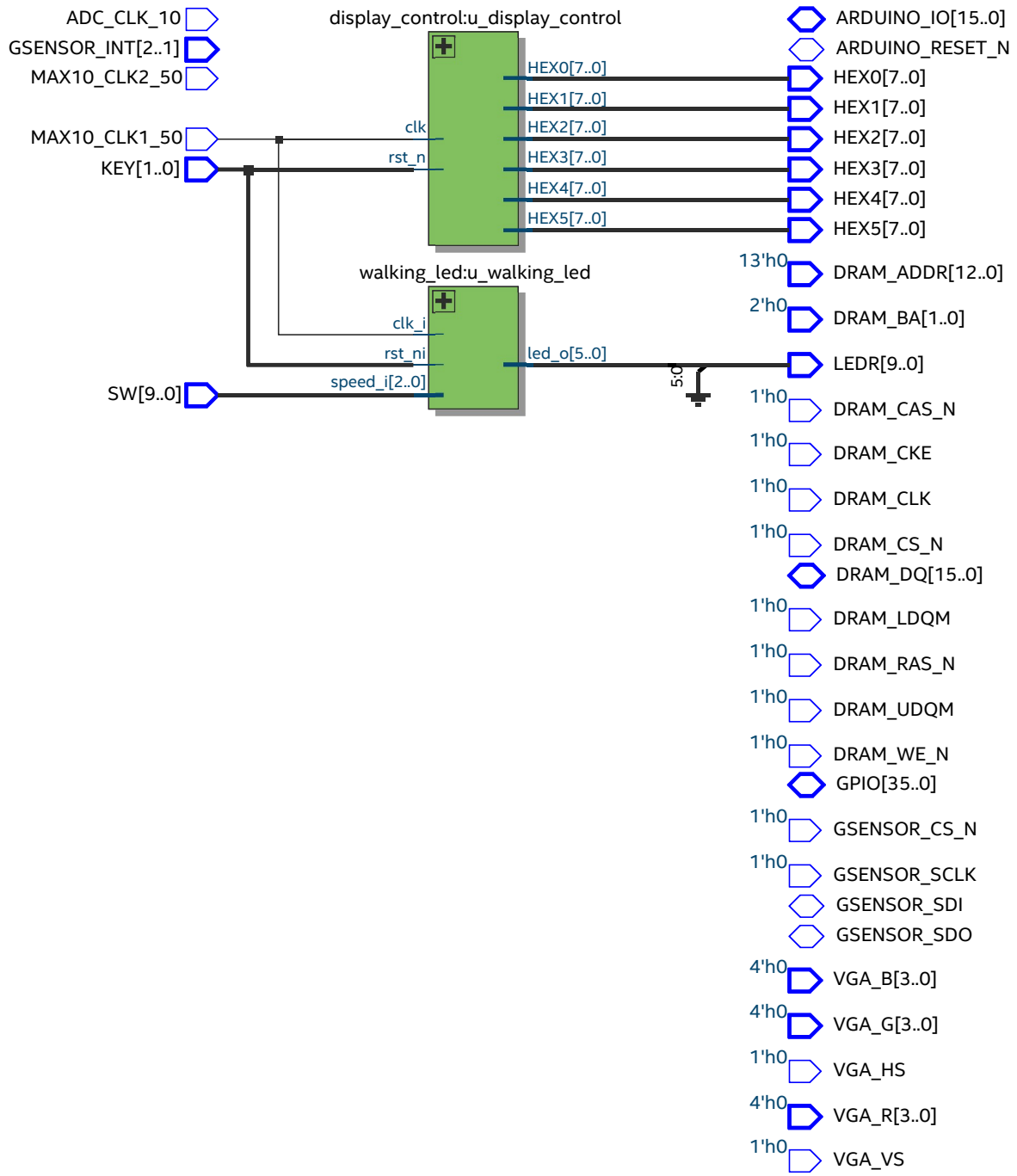
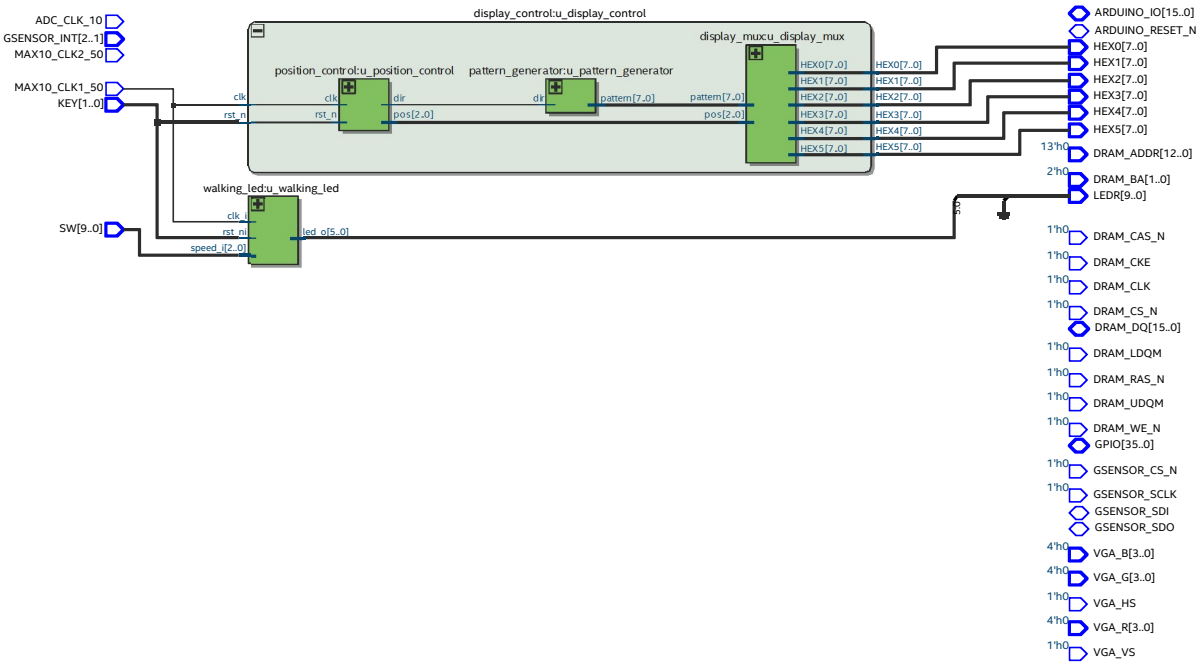
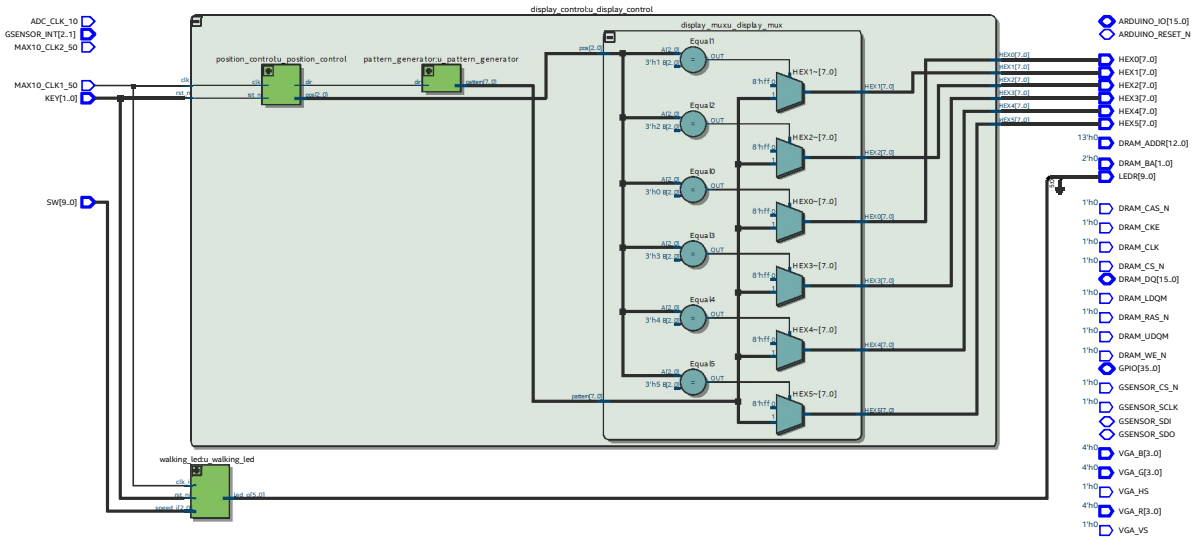
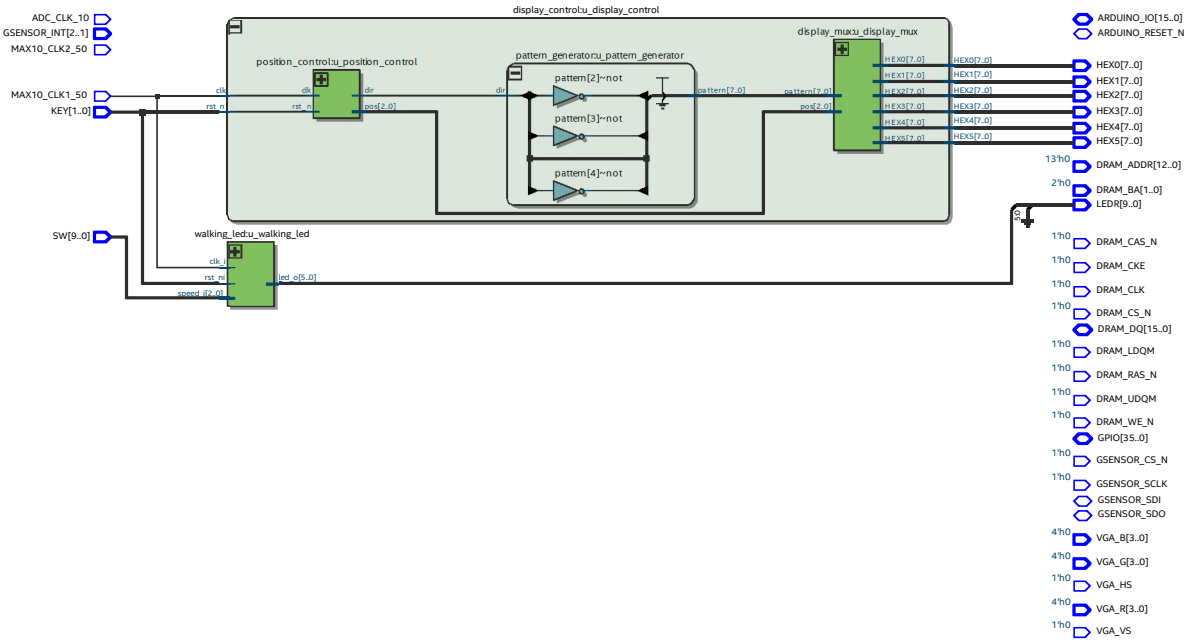
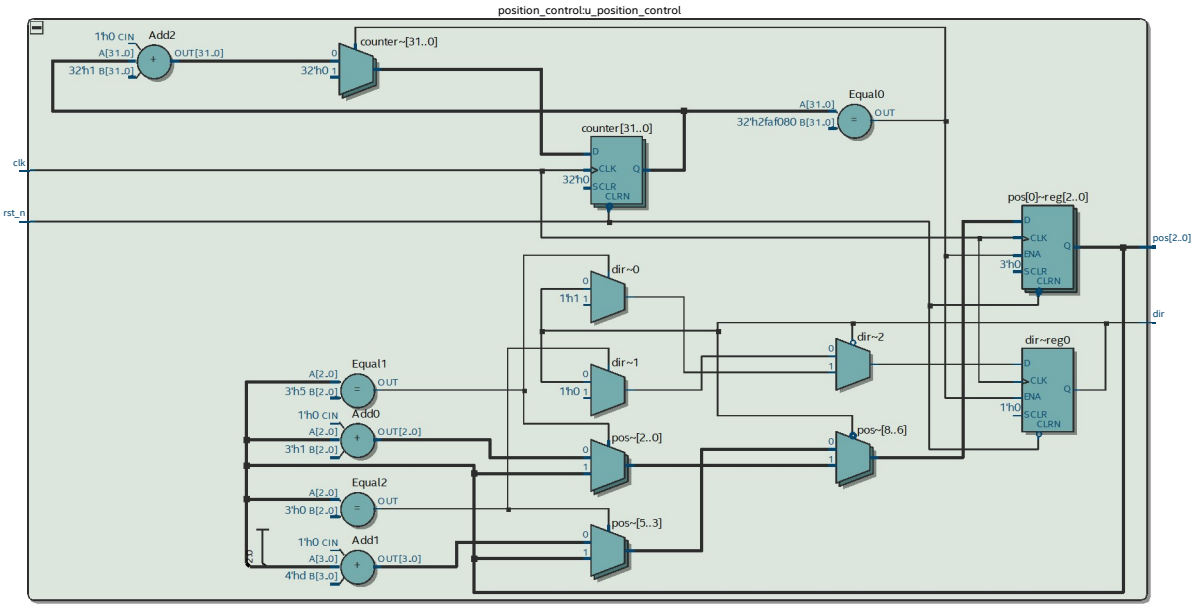
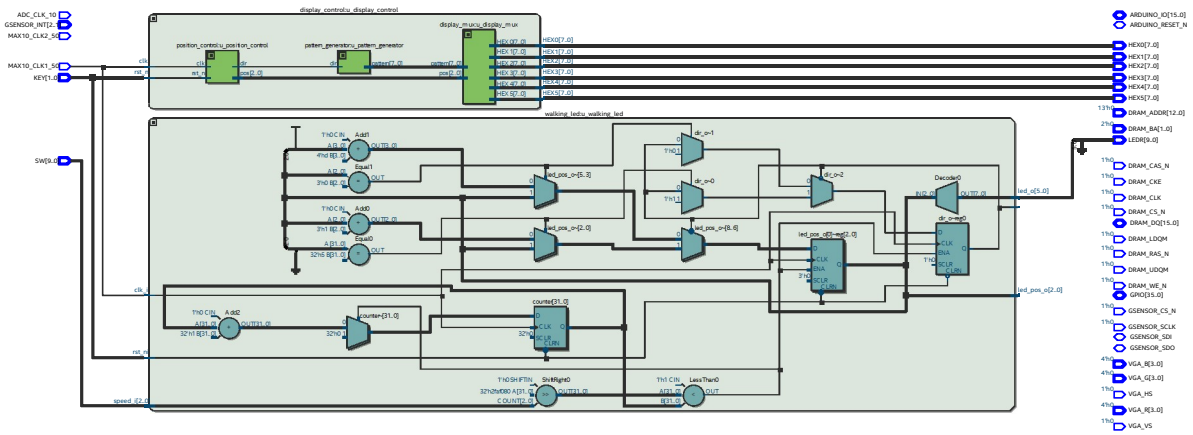

ADC_CLK_10
GSENSOR_INT[2..1]
MAX10_CLK2_50
MAX10_CLK1_50
KEY[1..0]
SW[9..0]
display_control:u_display_control
clk
rst_n
HEX0[7..0]
HEX1[7..0]
HEX2[7..0]
HEX3[7..0]
HEX4[7..0]
HEX5[7..0]
walking_led:u_walking_led
clk_i
rst_ni
speed_i[2..0]
led_o[5..0]
5:0
ARDUINO_IO[15..0]
ARDUINO_RESET_N
HEX0[7..0]
HEX1[7..0]
HEX2[7..0]
HEX3[7..0]
HEX4[7..0]
HEX5[7..0]
13'h0
DRAM_ADDR[12..0]
2'h0
DRAM_BA[1..0]
LEDR[9..0]
1'h0
DRAM_CAS_N
1'h0
DRAM_CKE
1'h0
DRAM_CLK
1'h0
DRAM_CS_N
DRAM_DQ[15..0]
1'h0
DRAM_LDQM
1'h0
DRAM_RAS_N
1'h0
DRAM_UDQM
1'h0
DRAM_WE_N
GPIO[35..0]
1'h0
GSENSOR_CS_N
1'h0
GSENSOR_SCLK
GSENSOR_SDI
GSENSOR_SDO
4'h0
VGA_B[3..0]
4'h0
VGA_G[3..0]
1'h0
VGA_HS
4'h0
VGA_R[3..0]
1'h0
VGA_VS


ADC_CLK_10
GSENSOR_INT[2..1]
MAX10_CLK2_50
MAX10_CLK1_50
KEY[1..0]
SW[9..0]
display_control:u_display_control
position_control:u_position_control
pattern_generator:u_pattern_generator
display_mux:u_display_mux
walking_led:u_walking_led
clk
rst_n
dir
pos[2..0]
pattern[7..0]
HEX0[7..0]
HEX1[7..0]
HEX2[7..0]
HEX3[7..0]
HEX4[7..0]
HEX5[7..0]
led_o[5..0]
speed_i[2..0]
ARDUINO_IO[15..0]
ARDUINO_RESET_N
13'h0
2'h0
1'h0
4'h0
DRAM_ADDR[12..0]
DRAM_BA[1..0]
LEDR[9..0]
DRAM_CAS_N
DRAM_CKE
DRAM_CLK
DRAM_CS_N
DRAM_DQ[15..0]
DRAM_LDQM
DRAM_RAS_N
DRAM_UDQM
DRAM_WE_N
GPIO[35..0]
GSENSOR_CS_N
GSENSOR_SCLK
GSENSOR_SDI
GSENSOR_SDO
VGA_B[3..0]
VGA_G[3..0]
VGA_HS
VGA_R[3..0]
VGA_VS


display_control:u_display_control
display_muxu_display_mux
position_control:u_position_control
pattern_generator:u_pattern_generator
walking_led:u_walking_led
ADC_CLK_10
GSENSOR_INT[2..1]
MAX10_CLK2_50
MAX10_CLK1_50
KEY[1..0]
SW[9..0]
Equal1
Equal2
Equal0
Equal3
Equal4
Equal5
HEX1~[7..0]
HEX2~[7..0]
HEX0~[7..0]
HEX3~[7..0]
HEX4~[7..0]
HEX5~[7..0]
ARDUINO_IO[15..0]
ARDUINO_RESET_N
HEX0[7..0]
HEX1[7..0]
HEX2[7..0]
HEX3[7..0]
HEX4[7..0]
HEX5[7..0]
DRAM_ADDR[12..0]
DRAM_BA[1..0]
LEDR[9..0]
DRAM_CAS_N
DRAM_CKE
DRAM_CLK
DRAM_CS_N
DRAM_DQ[15..0]
DRAM_LDQM
DRAM_RAS_N
DRAM_UDQM
DRAM_WE_N
GPIO[35..0]
GSENSOR_CS_N
GSENSOR_SCLK
GSENSOR_SDI
GSENSOR_SDO
VGA_B[3..0]
VGA_G[3..0]
VGA_HS
VGA_R[3..0]
VGA_VS


display_control:u_display_control
position_control:u_position_control
pattern_generator:u_pattern_generator
display_mux:u_display_mux
walking_led:u_walking_led
pattern[2]~not
pattern[3]~not
pattern[4]~not
ADC_CLK_10
GSENSOR_INT[2..1]
MAX10_CLK2_50
MAX10_CLK1_50
KEY[1..0]
SW[9..0]
ARDUINO_IO[15..0]
ARDUINO_RESET_N
HEX0[7..0]
HEX1[7..0]
HEX2[7..0]
HEX3[7..0]
HEX4[7..0]
HEX5[7..0]
DRAM_ADDR[12..0]
DRAM_BA[1..0]
LEDR[9..0]
DRAM_CAS_N
DRAM_CKE
DRAM_CLK
DRAM_CS_N
DRAM_DQ[15..0]
DRAM_LDQM
DRAM_RAS_N
DRAM_UDQM
DRAM_WE_N
GPIO[35..0]
GSENSOR_CS_N
GSENSOR_SCLK
GSENSOR_SDI
GSENSOR_SDO
VGA_B[3..0]
VGA_G[3..0]
VGA_HS
VGA_R[3..0]
VGA_VS


position_control:u_position_control
Add2
counter~[31..0]
Equal0
counter[31..0]
clk
rst_n
pos[0]~reg[2..0]
pos[2..0]
dir~0
dir
dir~2
dir~reg0
dir~1
Equal1
Add0
pos~[2..0]
pos~[8..6]
Equal2
pos~[5..3]
Add1


display_control:u_display_control
position_control:u_position_control
pattern_generator:u_pattern_generator
display_mux:u_display_mux
walking_led:u_walking_led
ADC_CLK_10
GSENSOR_INT[2..
MAX10_CLK2_50
MAX10_CLK1_50
KEY[1..0]
SW[9..0]
HEX0[7..0]
HEX1[7..0]
HEX2[7..0]
HEX3[7..0]
HEX4[7..0]
HEX5[7..0]
DRAM_ADDR[12..0]
DRAM_BA[1..0]
LEDR[9..0]
DRAM_CAS_N
DRAM_CKE
DRAM_CLK
DRAM_CS_N
DRAM_DQ[15..0]
DRAM_LDQM
DRAM_RAS_N
DRAM_UDQM
DRAM_WE_N
GPIO[35..0]
GSENSOR_CS_N
GSENSOR_SCLK
GSENSOR_SDI
GSENSOR_SDO
VGA_B[3..0]
VGA_G[3..0]
VGA_HS
VGA_R[3..0]
VGA_VS
ARDUINO_IO[15..0]
ARDUINO_RESET_N
Add1
Equal1
Add0
Equal0
Add2
counter[31..0]
LessThan0
ShiftRight0
Decoder0
led_pos_o[5..3]
led_pos_o[2..0]
led_pos_o[8..6]
led_o[5..0]

```
//=======================================================
//  DE10-Lite Top-Level Module
//=======================================================

`define ENABLE_ADC_CLOCK
`define ENABLE_CLOCK1
`define ENABLE_CLOCK2
`define ENABLE_SDRAM
`define ENABLE_HEX0
`define ENABLE_HEX1
`define ENABLE_HEX2
`define ENABLE_HEX3
`define ENABLE_HEX4
`define ENABLE_HEX5
`define ENABLE_KEY
`define ENABLE_LED
`define ENABLE_SW
`define ENABLE_VGA
`define ENABLE_ACCELEROMETER
`define ENABLE_ARDUINO
`define ENABLE_GPIO

module DE10_LITE_Golden_Top(

    ///////////// ADC CLOCK: 3.3-V LVTTL //////////
`ifdef ENABLE_ADC_CLOCK
    input ADC_CLK_10,
`endif

    ///////////// CLOCK 1: 3.3-V LVTTL //////////
`ifdef ENABLE_CLOCK1
    input MAX10_CLK1_50,
`endif

    ///////////// CLOCK 2: 3.3-V LVTTL //////////
`ifdef ENABLE_CLOCK2
    input MAX10_CLK2_50,
`endif

    ///////////// SDRAM: 3.3-V LVTTL //////////
`ifdef ENABLE_SDRAM
    output [12:0] DRAM_ADDR,
    output [1:0] DRAM_BA,
    output DRAM_CAS_N,
    output DRAM_CKE,
    output DRAM_CLK,
    output DRAM_CS_N,
    inout [15:0] DRAM_DQ,
    output DRAM_LDQM,
    output DRAM_RAS_N,
    output DRAM_UDQM,
    output DRAM_WE_N,
`endif

    ///////////// SEG7: 3.3-V LVTTL //////////
`ifdef ENABLE_HEX0
    output [7:0] HEX0,
`endif
`ifdef ENABLE_HEX1
    output [7:0] HEX1,
`endif
`ifdef ENABLE_HEX2
    output [7:0] HEX2,
`endif
`ifdef ENABLE_HEX3
    output [7:0] HEX3,
`endif
`ifdef ENABLE_HEX4
    output [7:0] HEX4,
`endif
`ifdef ENABLE_HEX5
    output [7:0] HEX5,
`endif
```

```
74
75      //////////// KEY: 3.3 V SCHMITT TRIGGER //////////
76    `ifdef ENABLE_KEY
77      input [1:0] KEY,
78    `endif
79
80      //////////// LED: 3.3-V LVTTL //////////
81    `ifdef ENABLE_LED
82      output [9:0] LEDR,
83    `endif
84
85      //////////// SW: 3.3-V LVTTL //////////
86    `ifdef ENABLE_SW
87      input [9:0] SW,
88    `endif
89
90      //////////// VGA: 3.3-V LVTTL //////////
91    `ifdef ENABLE_VGA
92      output [3:0] VGA_B,
93      output [3:0] VGA_G,
94      output VGA_HS,
95      output [3:0] VGA_R,
96      output VGA_VS,
97    `endif
98
99      //////////// Accelerometer: 3.3-V LVTTL //////////
100   `ifdef ENABLE_ACCELEROMETER
101     output GSENSOR_CS_N,
102     input [2:1] GSENSOR_INT,
103     output GSENSOR_SCLK,
104     inout GSENSOR_SDI,
105     inout GSENSOR_SDO,
106   `endif
107
108     //////////// Arduino: 3.3-V LVTTL //////////
109   `ifdef ENABLE_ARDUINO
110     inout [15:0] ARDUINO_IO,
111     inout ARDUINO_RESET_N,
112   `endif
113
114     //////////// GPIO, GPIO connect to GPIO Default: 3.3-V LVTTL //////////
115   `ifdef ENABLE_GPIO
116     inout [35:0] GPIO
117   `endif
118   );
119
120   //=======================================================
121   //  Wire Declarations
122   //=======================================================
123
124   wire [2:0] walking_col, col;
125   wire walking_row, row;
126
127   //=======================================================
128   //  Instantiere module
129   //=======================================================
130
131   // Modulul pentru controlul display-ului 7-segment
132   display_control u_display_control (
133       .clk(MAX10_CLK1_50),
134       .rst_n(KEY[0]), // Reset activ pe 0
135       .HEX0(HEX0),
136       .HEX1(HEX1),
137       .HEX2(HEX2),
138       .HEX3(HEX3),
139       .HEX4(HEX4),
140       .HEX5(HEX5)
141   );
142
143   // SW[9] selecteaza modul de operare:
144   // 0: Pozitia simbolului este controlata manual prin SW[3:0]
145   // 1: Simbolul se misca automat
146   assign row = SW[9] ? walking_row : SW[3];
```

```
147   assign col = SW[9] ? walking_col : SW[2:0];
148
149   // Modul care simuleaza un LED care se misca pe afisaje
150   walking_led #(
151       .CLK_FREQ_HZ(50_000_000),  // 50 MHz
152       .NO_LED(6),                // 6 pozitii pe display
153       .STEP_PERIOD_S(1)          // Perioada de miscare: 1 secunda
154   ) u_walking_led (
155       .clk_i(MAX10_CLK1_50),
156       .rst_ni(KEY[0]),
157       .speed_i(SW[8:6]),    // Viteza controlata de SW[8:6]
158       .led_o(LEDR[5:0]),    // Output catre LED-uri
159       .led_pos_o(walking_col),
160       .dir_o(walking_row)
161   );
162
163   endmodule
164
```

```
1   module display_control (
2       input clk,
3       input rst_n,
4       output [7:0] HEX0,
5       output [7:0] HEX1,
6       output [7:0] HEX2,
7       output [7:0] HEX3,
8       output [7:0] HEX4,
9       output [7:0] HEX5
10  );
11
12  wire [2:0] pos;  // Pozitia simbolului
13  wire dir;        // Directia
14  wire [7:0] pattern; // Simbolul afisat
15
16  // Instantierea modulelor
17  position_control u_position_control (
18      .clk(clk),
19      .rst_n(rst_n),
20      .pos(pos),
21      .dir(dir)
22  );
23
24  pattern_generator u_pattern_generator (
25      .dir(dir),
26      .pattern(pattern)
27  );
28
29  display_mux u_display_mux (
30      .pos(pos),
31      .pattern(pattern),
32      .HEX0(HEX0),
33      .HEX1(HEX1),
34      .HEX2(HEX2),
35      .HEX3(HEX3),
36      .HEX4(HEX4),
37      .HEX5(HEX5)
38  );
39
40  endmodule
41
```

```
1   module display_mux (
2       input [2:0] pos,       // Pozitia simbolului (0-5)
3       input [7:0] pattern,   // Simbolul curent
4       output reg [7:0] HEX0, // Afisajele 7-seg
5       output reg [7:0] HEX1,
6       output reg [7:0] HEX2,
7       output reg [7:0] HEX3,
8       output reg [7:0] HEX4,
9       output reg [7:0] HEX5
10  );
11
12  localparam EMPTY = 8'b11111111; // Valoare pentru afisaj gol
13
14  always @(*) begin
15      HEX0 = (pos == 3'b000) ? pattern : EMPTY;
16      HEX1 = (pos == 3'b001) ? pattern : EMPTY;
17      HEX2 = (pos == 3'b010) ? pattern : EMPTY;
18      HEX3 = (pos == 3'b011) ? pattern : EMPTY;
19      HEX4 = (pos == 3'b100) ? pattern : EMPTY;
20      HEX5 = (pos == 3'b101) ? pattern : EMPTY;
21  end
22
23  endmodule
24
```

```
1   module pattern_generator (
2       input dir,            // Directia de miscare
3       output reg [7:0] pattern // Simbolul pentru afisaj
4   );
5
6   localparam UP = 8'b10011100;   // Simbolul pentru UP
7   localparam DOWN = 8'b10100011; // Simbolul pentru DOWN
8
9   always @(*) begin
10      pattern = dir ? DOWN : UP;
11  end
12
13  endmodule
14
```

```
1   module position_control (
2       input clk,        // Semnal de ceas
3       input rst_n,      // Reset activ pe 0
4       output reg [2:0] pos, // Pozitia curenta (0-5)
5       output reg dir    // Directia (0 = dreapta, 1 = stanga)
6   );
7
8   reg [31:0] counter; // Contor pentru generarea intarzierii
9
10  always @(posedge clk or negedge rst_n) begin
11      if (!rst_n) begin
12          pos <= 3'b000; // Porneste de la pozitia 0
13          dir <= 1'b0;   // Merge spre dreapta initial
14          counter <= 32'b0;
15      end
16      else if (counter == 32'd50000000) begin  // Dupa 1 secunda
17          counter <= 32'b0;
18          if (dir == 1'b0) begin  // Daca merge spre dreapta
19              if (pos == 3'b101) begin
20                  dir <= 1'b1; // Schimba directia spre stanga
21              end
22              else begin
23                  pos <= pos + 1;
24              end
25          end
26          else begin  // Daca merge spre stanga
27              if (pos == 3'b000) begin
28                  dir <= 1'b0; // Schimba directia spre dreapta
29              end
30              else begin
31                  pos <= pos - 1;
32              end
33          end
34      end
35      else begin
36          counter <= counter + 1;
37      end
38  end
39
40  endmodule
41
```

```
module walking_led #(
    parameter CLK_FREQ_HZ = 50_000_000,  // Frecventa clock-ului (50 MHz)
    parameter NO_LED = 6,                // Numar de pozitii pe display
    parameter STEP_PERIOD_S = 1          // Perioada de miscare (1 secunda)
)(
    input clk_i,         // Clock principal
    input rst_ni,        // Reset activ LOW
    input [2:0] speed_i, // Controlul vitezei prin switch-uri
    output reg [NO_LED-1:0] led_o, // LED-urile active
    output reg [2:0] led_pos_o,    // Pozitia LED-ului activ
    output reg dir_o               // Directia de miscare (0 = dreapta, 1 = stanga)
);

localparam COUNTER_MAX = CLK_FREQ_HZ * STEP_PERIOD_S; // Numar de cicluri pentru un pas
reg [31:0] counter; // Contor intern

always @(posedge clk_i or negedge rst_ni) begin
    if (!rst_ni) begin
        led_pos_o <= 3'b000;
        dir_o <= 1'b0;
        counter <= 0;
    end
    else begin
        if (counter >= COUNTER_MAX >> speed_i) begin // Controlul vitezei
            counter <= 0;

            if (dir_o == 1'b0) begin
                if (led_pos_o == NO_LED - 1)
                    dir_o <= 1'b1; // Schimba directia
                else
                    led_pos_o <= led_pos_o + 1;
            end
            else begin
                if (led_pos_o == 3'b000)
                    dir_o <= 1'b0; // Schimba directia
                else
                    led_pos_o <= led_pos_o - 1;
            end
        end
        else begin
            counter <= counter + 1;
        end
    end
end

always @(*) begin
    led_o = 6'b000000;
    led_o[led_pos_o] = 1'b1; // Activeaza LED-ul corespunzator pozitiei
end

endmodule

```