Lab7_1

задание для самостоятельного выполнения

Плата для аппаратной отладки проекта

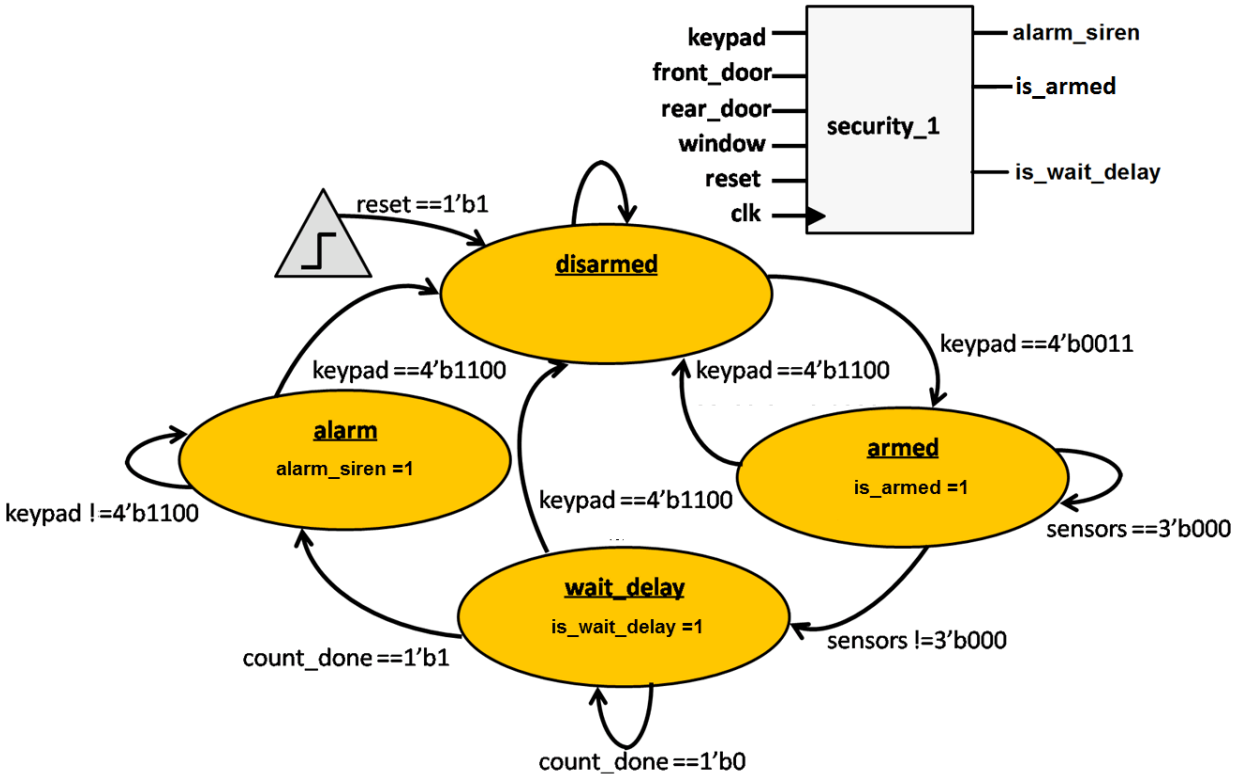
Аппаратная отладка проекта ориентирована на плату MiniDiLaB-CIV

MiniDiLaB-CIV: Микросхема - **EP4CE6E22C8**, Вход тактового сигнала (25МГц) – 23.

Описание проекта Lab7_1

Рабочая папка Lab7_1. Имя проекта – Lab7_1. Имя модуля верхнего уровня – Lab7_1. Файл с описанием – Lab7_1.sv. Файл теста – tb_Lab7_1.sv. Файл с описанием для отладки – db_Lab7_1.sv.

Алгоритм работы:



The signals in the above figure are:

Inputs

- keypad[3:0] – 4-bit input used to arm (0011) or disarm (1100) the security system.
- sensors[2:0] – 3-bit internal signal formed by concatenating the inputs - front_door, rear_door and window. If any of the 3 bits in this signal goes high, the alarm would be triggered after a delay of 100 clock cycles.
  - front_door, rear_door, window – Single-bit inputs which are assumed to go high when security is breached and the alarm should be activated.
- CLK – Master clock signal.
- reset – System reset (synchronous).

Outputs:

- alarm_siren – Output which indicates that the system is in 'alarm' state (i.e, the alarm has been activated).
- is_armed – Output which indicates that the system is in 'armed' state (i.e, the security system is on).
- is_wait_delay – Output which indicates that the system is in 'wait_delay' state (i.e, a security breach has been detected and the system is waiting for 100 clock cycles before activating the alarm).
- start_count – Internal signal for counting 100 clock cycles before triggering the alarm.
- count_done – Internal signal that activates the alarm after 100 clock cycles have been counted.

Программа работы

- Разработать описание конечного автомата – модуль Lab7_1, с использованием расширений SystemVerilog (типов данных, конструкций…)
    - ***К описанию конечного автомата надо добавить*** вход ENA, разрешающий (=1)/запрещающий (=0) работу автомата.
    - ***К описанию конечного автомата надо добавить*** счетчик делитель, обеспечивающий:
        - При реализации отладки конечного автомата на плате: деление входного тактового сигнала CLK (25МГц) так, что бы сигнал переноса, поступающий на вход ENA конечного автомата, имел частоту 10Гц (за одну секунду – 10 импульсов), т.е. частоту 25МГц надо поделить на 2 500 000.
        - При моделировании: деление на 4.
- Разработать тест tb_Lab7_1 для проверки конечного автомата Lab7_1 с использованием расширений SystemVerilog (тест первого класса – без автоматической проверки).
- По результатам моделирования в ModelSim необходимо доказать работоспособность конечного автомата (продемонстрировать переход в каждое состояние, использование всех ребер, выдерживание задержки). Тест первого уровня (без автоматической проверки).
- Разработать модуль верхнего уровня для отладки db_Lab7_1, содержащий: модуль Lab7_1; модуль SP_unit (модуль, обеспечивающий возможность: задания входных управляющих сигналов без использования кнопок на плате; отображения состояний автомата и выходных сигналов). Модуль должен обеспечивать подключение к тактовому сигналу на плате.
- Настроить логический анализатор для проведения исследования и отладки реализуемого на плате db_Lab7_1.
- Провести анализ работы db_Lab7_1 и доказать (зафиксировав результаты снимками экрана), что:
    - Модуль управляется входными сигналами
    - Правильно реализуется алгоритм работы

Содержание отчета

- Отчет должен быть оформлен по правилам, принятым в Высшей Школе.
- Отчет должен содержать все этапы работы, все созданные исходные коды, необходимые снимки экрана. Все рисунки и полученные на них результаты должны быть прокомментированы.

Пример кода

Ниже приведен пример кода для описания конечного автомата, созданный с использованием подмножеств Verilog95 и Verilog 2001.

Пример приведен для упрощения разработки описания конечного автомата с использованием расширений SystemVerilog

```
1   `timescale 1ns / 1ps
2   module security_verilog(
3       input front_door,
4       input rear_door,
5       input window,
6       input clk,
7       input reset,
8       input [3:0] keypad,
9       output reg alarm_siren,
10      output reg is_armed,
11      output reg is_wait_delay
12      );
13  // set the delay value (the number of clocks between a faulted zone and the
14  // alarm going off)
15  parameter         delay_val     = 100;
16  // Variables used for counting 100 (delay_val) clock cycles
17  wire start_count;
18  wire count_done;
19  reg [6:0] delay_cntr = 0 ;
20  // Max value of delay_cntr is delay_val (i.e., d'100 or b'1100100)
21  localparam        disarmed    =  2'd0,
22                    armed       =  2'd1,
23                    wait_delay  =  2'd2,
24                    alarm       =  2'd3;
25  reg [1:0] curr_state, next_state ;
26  wire [2:0] sensors ; // used to combine inputs
27  assign sensors = { front_door, rear_door, window } ;
28  //  procedural block for incrementing the state machine
29  always @ ( posedge clk )
30    if (reset)
31      curr_state <= disarmed ;
32    else
33      curr_state <= next_state ;
34  // procedural block to determine the next state
35  always @ ( curr_state, sensors, keypad, count_done ) begin
36    case ( curr_state )
37      disarmed: begin
38        if ( keypad == 4'b0011 )
39          next_state <= armed;
40        else
41          next_state <= curr_state ;
42      end
43
44      armed: begin
45        if ( sensors != 3'b000 )
46          next_state <= wait_delay;
47        else if ( keypad == 4'b1100)
48          next_state <= disarmed;
49        else
50          next_state <= curr_state ;
51      end
```

```
52
53          wait_delay: begin
54            if (count_done == 1'b1)
55              next_state <= alarm;
56            else if ( keypad == 4'b1100 )
57              next_state <= disarmed ;
58            else
59              next_state <= curr_state ;
60          end
61
62          alarm: begin
63            if ( keypad == 4'b1100 )
64              next_state <= disarmed;
65            else
66              next_state <= curr_state ;
67          end
68        endcase
69      end
70      // procedural block to generate the state machine output values
71       always @ ( posedge clk ) begin
72         if (reset) begin
73           is_armed      <= 1'b0 ;
74           is_wait_delay <= 1'b0 ;
75           alarm_siren   <= 1'b0 ;
76         end
77         else
78         begin
79           is_armed      <= ( next_state == armed );
80           is_wait_delay <= ( next_state == wait_delay );
81           alarm_siren   <= ( next_state == alarm ) ;
82         end
83       end
84       assign start_count = (( curr_state == armed) && (sensors != 3'b000));
85      // Implement the delay counter.
86      // Loads delay_cntr with delay_val-1 when start_count is high, then counts
87      // down to 0 and stops.
88      // The condition delay_cntr = 0 triggers the next state transition in the
89      // main state machine
90       always @ ( posedge clk) begin
91       if (reset)
92         delay_cntr <= 0;
93       else if (start_count)
94         delay_cntr <= delay_val - 1'b1;
95       else if (curr_state != wait_delay)
96         delay_cntr <= 0;
97       else if (delay_cntr != 0)
98         delay_cntr <= delay_cntr - 1'b1;
99       end
100
101      assign count_done = (delay_cntr == 0);
102    endmodule
```