lab_MS_SV2

для самостоятельного выполнения

Платы для аппаратной отладки проекта

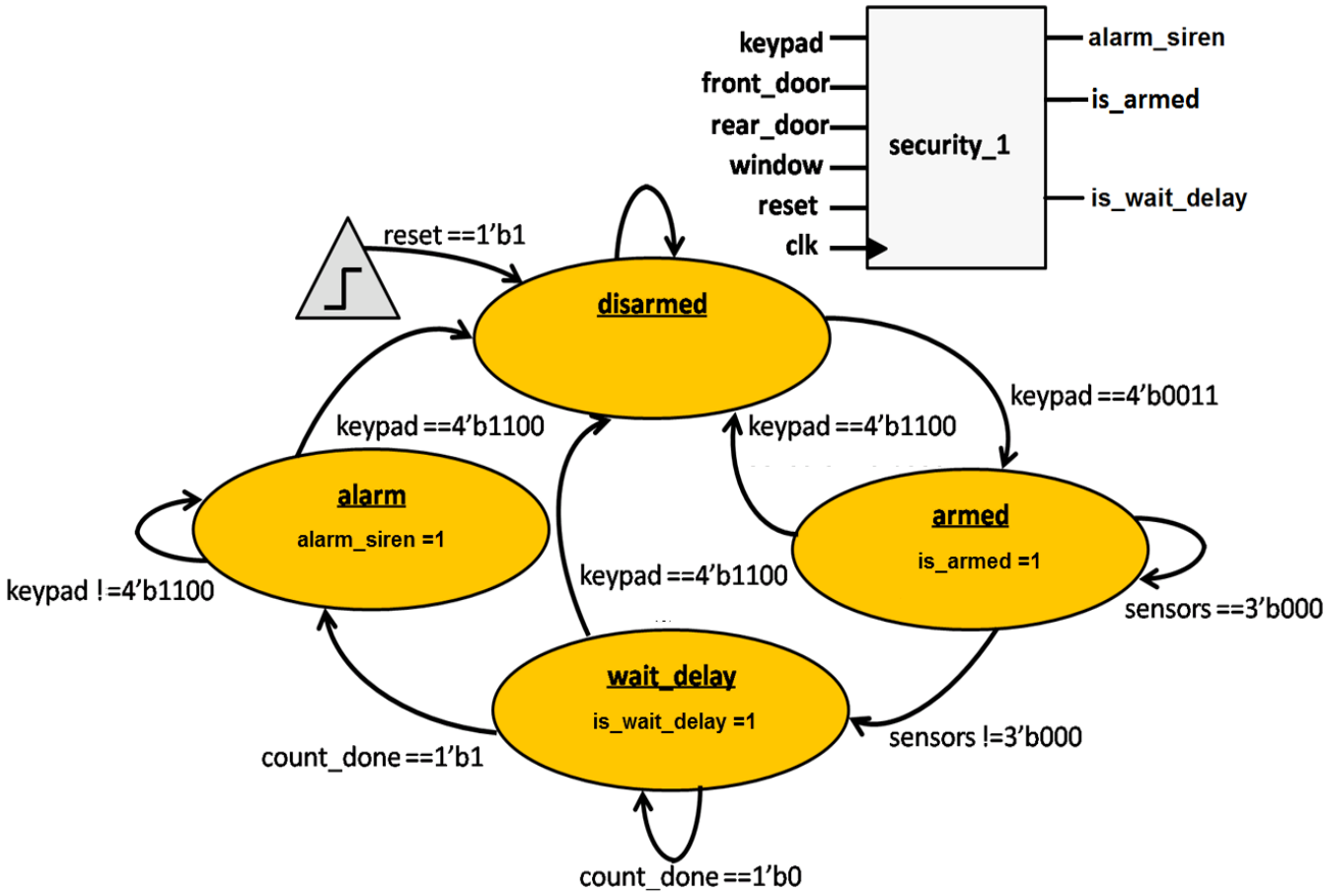
- **Cyclone IV E**
  - **EP4CE6E22C8 –** для платы **MiniDilab-CIV**
- **MAX10**
  - **10M50DAF484C6GES** – для платы **MAX10 NEEK**

Имя проекта, папки и файлы

Рабочая папка C:\Intel_trn\Q_MS_SV\lab_MS_SV2.
Имя проекта – lab_MS_SV2.
Имя модуля верхнего уровня – lab_MS_SV2.
Файл с описанием – lab_MS_SV2.sv.
Файл теста – tb_ lab_MS_SV2.sv.
Файл с описанием для отладки – db_lab_MS_SV2.sv.
Файл для реализации – impl_lab_MS_SV2.sv.

Описание проекта

Алгоритм работы:



Входы

- CLK – вход тактового сигнал
- reset - вход синхронного сброса (активный уровень – 1)
- keypad[3:0] – код 4-бита: включение системы – код 0011; выключение системы – код 1100.
- sensors[2:0] – вход сенсоров (3-бита):
  - front_door – входная дверь,
  - rear_door – задняя дверь,
  - window – окно.

  Если любой из трех входов становится равен 1 (т.е. «открыт»), то сигнализация должна сработать через время, задаваемое задержкой в 100 тактов сигнала CLK

Выходы:

- alarm_siren – единица на выходе означает срабатывание сигнализации.
- is_armed – единица на выходе означает, что система находится во включенном состоянии.
- is_wait_delay – единица на выходе означает, что система находится в режиме ожидания. (ожидание 100 тактов сигнала CLK от момента срабатывания сенсоров.)
  - count_done – внутренний сигнал окончания задержки в 100 тактов сигнала CLK (= 1 – прошло 100 тактов).

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
```

Программа работы

- Разработать (с использованием расширений SystemVerilog) описание конечного автомата – модуль **lab_MS_SV2**
  - ***К описанию конечного автомата, приведенному в примере, надо добавить*** вход ENA, разрешающий (=1)/запрещающий (=0) работу автомата и формирователя задержки в 100 тактовых сигналов.
- Разработать (с использованием расширений SystemVerilog) тест **tb_lab_MS_SV2** для проверки конечного автомата **lab_MS_SV2** (тест первого класса – без автоматической проверки).
  - По результатам моделирования в ModelSim необходимо доказать работоспособность конечного автомата (продемонстрировать переход в каждое состояние, использование всех ребер, формирование задержки в 100 тактов).
- В пакете Quartus на базе IP модуля In-System Source and Probe создать модуль SP_unit, обеспечивающий:
  - возможность задания входных управляющих сигналов для модуля **lab_MS_SV2** без использования кнопок на плате.
    - формируемые управляющие сигналы, должны быть привязаны к тактовому сигналу CLK.
- Разработать (с использованием расширений SystemVerilog) модуль верхнего уровня для отладки **db_lab_MS_SV2**, содержащий:
  - модуль **lab_MS_SV2**;
  - модуль **SP_unit**.
  - Вход CLK, подключенный к тактовому сигналу на плате

```
(* altera_attribute = "-name IO_STANDARD \"3.3-V LVCMOS\"", chip_pin = "R8"  *)
//"23" for miniDilab-CIV
//"R8" for DE0_nano
//"N5" for MAX10 NEEK
input CLK
```

- Настроить логический анализатор для проведения исследования и отладки на плате модуля **db_lab_MS_SV2**.
  - 1 сегмент, 128 отсчетов, pre-trigger position
  - Trigger Conditions - 1
    - Условие (Trigger Condition 1): keypad[3:0] =1100 и любой (или несколько) из сенсоров =1
- Провести проверку работы модуля **db_lab_MS_SV2** на плате.
- Разработать модуль для реализации **impl_lab_MS_SV2**:
  - Модуль должен содержать lab**_MS_SV2**
  - ***К описанию модуля надо добавить***
    - счетчик делитель, обеспечивающий:
      деление входного тактового сигнала CLK так, чтобы сигнал переноса, поступающий с выхода счетчика-делителя на вход ENA конечного автомата, имел частоту 10Гц (за одну секунду – 10 импульсов), т.е. частоту 25МГц надо поделить на 2 500 000 (плата **miniDiLaB-IV**), частоту 50МГц надо поделить на 5 000 000 (плата **MAX10 NEEK**).
  - Все входы, включая вход **reset**, надо подключить к **переключателям** платы.
    - На всех входах должно быть по 2 последовательных регистра, синхронизируемых сигналом CLK.
  - Все выходы (**alarm_siren, is_armed, is_wait_delay**) надо подключить к светодиодам.
- Реализовать модуль **impl_lab_MS_SV2** на плате, проверить его работу на соответствие алгоритму и показать преподавателю.

Содержание отчета

- Отчет должен содержать все этапы работы, все созданные исходные коды, необходимые снимки экрана. Все рисунки и полученные на них результаты должны быть прокомментированы.