Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и кибербезопасности
Высшая школа компьютерных технологий и информационных систем

Отчёт по лабораторной работе № 4

Дисциплина: Автоматизация проектирования дискретных устройств (на английском языке).

Выполнил студент гр. 5130901/10101 _____ Д.Л. Симоновский
(подпись)

Руководитель _____ А.А. Федотов
(подпись)

“28” февраля 2024 г.

Санкт-Петербург
2024

Оглавление

1. Список иллюстраций:

2. Цель упражнения:

Пройти цикл проектирования в рамках пакетов Quartus и ModelSim, включая следующие этапы:

- Создание проекта.
- Разработка описания модулей с использованием конструкций расширения SystemVerilog.
- Разработка теста на языке SystemVerilog и моделирование.
- Отладка проекта.

3. Алгоритм работы проекта:

Алгоритм работы проекта приведен ниже:

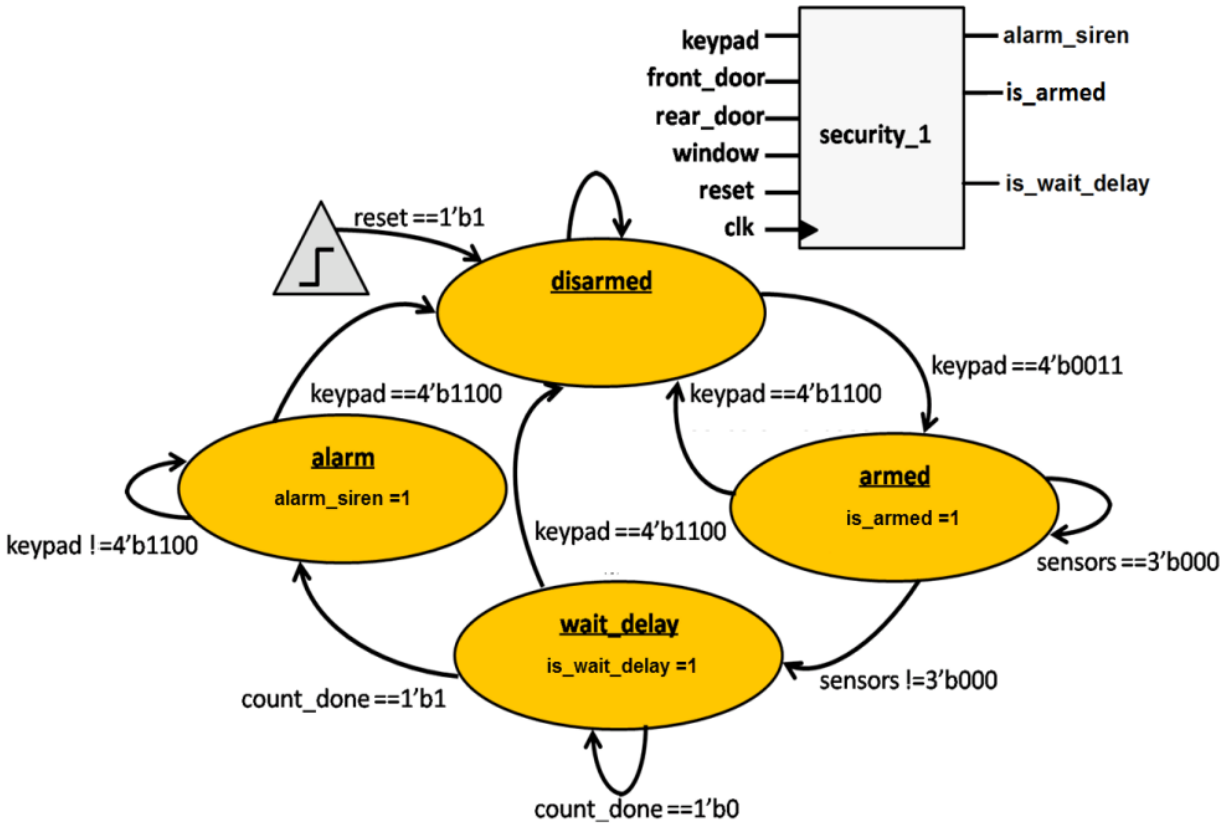

Рис. 3.1. Алгоритм работы разрабатываемого проекта.

К описанию конечного автомата, приведенному в примере, надо добавить вход ENA, разрешающий (=1) / запрещающий (=0) работу автомата и формирователя задержки в 100 тактовых сигналов.

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.)

4. Решение:

Создадим модуль lab_MS_SV2, задав входы и выходы типом данных bit, который позволит не думать о том, какой тип нужен net или variables:

Verilog_labs - lab_MS_SV2.sv

```
1  `timescale 1ns / 1ns
2  module lab_MS_SV2 (
3    input  bit       front_door,
4    input  bit       rear_door,
5    input  bit       window,
6    input  bit       clk,
7    input  bit       reset,
8    input  bit       ENA,
9    input  bit [3:0] keypad,
10   output bit       alarm_siren,
11   output bit       is_armed,
12   output bit       is_wait_delay
13 );
14
15 endmodule
```

Теперь необходимо реализовать конечный автомат в соответствии с поставленным заданием. Пусть состояния определяются используя комбинационную схему, а задаются, используя триггерную. Результат же пусть выводится с использованием триггерной схемы:

Verilog_labs - lab_MS_SV2.sv

```
19 enum bit [1:0] { disarmed, armed, wait_delay, alarm } curr_state, next_state;
20 bit [2:0] sensors;
21 assign sensors = {front_door, rear_door, window};
22
23 always_ff @(posedge clk) begin : go_to_next_state
24   if (ENA)
25     if (reset) curr_state <= disarmed;
26     else curr_state <= next_state;
27 end
28
29 always_comb begin : get_next_state
30   case (curr_state)
31     disarmed:
32       if (keypad == 4'b0011) next_state = armed;
33       else next_state = curr_state;
34     armed:
35       if (sensors != 3'b000) next_state = wait_delay;
36       else if (keypad == 4'b1100) next_state = disarmed;
37       else next_state = curr_state;
38     wait_delay:
39       if (count_done == 1'b1) next_state = alarm;
40       else if (keypad == 4'b1100) next_state = disarmed;
41       else next_state = curr_state;
42     alarm:
43       if (keypad == 4'b1100) next_state = disarmed;
44       else next_state = curr_state;
45   endcase
46 end
47
48 always_ff @(posedge clk) begin : set_output
49   if (ENA)
50     if (reset) {is_armed, is_wait_delay, alarm_siren} <= 1'b0;
51     else begin
52       is_armed      <= (curr_state == armed);
53       is_wait_delay <= (curr_state == wait_delay);
54       alarm_siren   <= (curr_state == alarm);
55     end
56 end
```

Осталось добавить счетчик на 100 единиц, который бы отсчитывал время между срабатыванием сенсора и перехода к тревоге:

Verilog_labs - lab_MS_SV2.sv

```
58  parameter delay_val = 100;
59
60  bit start_count, count_done;
61  bit [6:0] delay_cntr = 0;
62
63  assign start_count = ((curr_state == armed) && (sensors != 3'b000));
64
65  always_ff @(posedge clk) begin : counter_alarm
66    if (ENA)
67      if (reset) delay_cntr <= 0;
68      else if (start_count) delay_cntr <= delay_val - 1'b1;
69      else if (curr_state != wait_delay) delay_cntr <= 1'b0;
70      else if (delay_cntr != 0) delay_cntr <= delay_cntr - 1'b1;
71  end
72
73  assign count_done = (delay_cntr == 0);
```

Таким образом мы получили модуль в соответствии с заданием.
Его RTL схема приведена ниже:

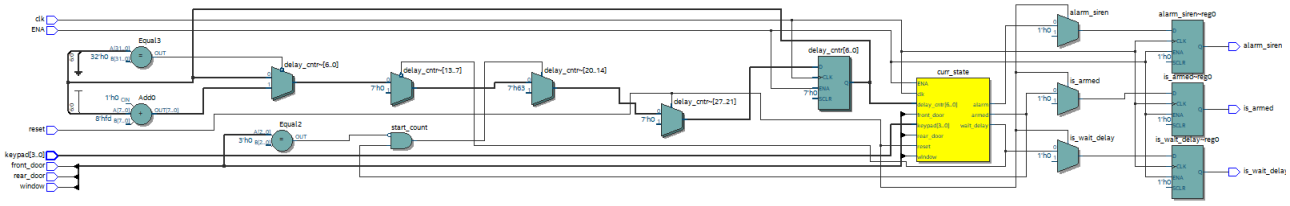

Рис. 4.1. RTL схема разработанного устройства.

А получившийся конечный автомат выглядит следующим образом:

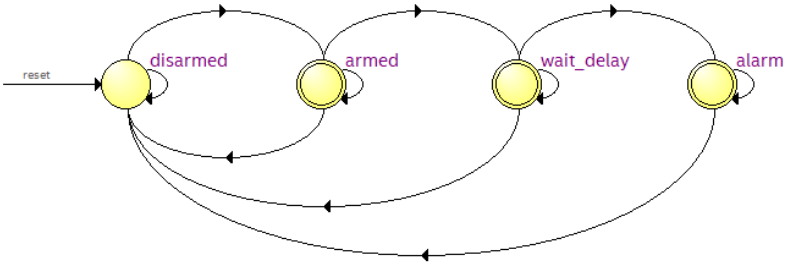

Рис. 4.2. Разработанный конечный автомат.

Теперь необходимо его протестировать, для этого разработаем тест первого класса, проверив все необходимые переходы.
Перейдем к созданию необходимых входов для модуля, а также clk. Стоит отметить, что SystemVerilog позволяет упростить задание входов и выходов для модуля:

Verilog_labs - tb_lab_MS_SV2.sv

```
1  `timescale 1ns / 1ns
2  module tb_lab_MS_SV2;
3    parameter PERIOD = 10;
4    bit       front_door = 0;
5    bit       rear_door = 0;
6    bit       window = 0;
7    bit       clk = 0;
8    bit       reset = 0;
9    bit       ENA = 0;
10   bit [3:0] keypad = 0;
11   bit       alarm_siren;
12   bit       is_armed;
13   bit       is_wait_delay;
14
15   initial forever #(PERIOD / 2) clk = ~clk;
16
17   lab_MS_SV2 u_lab_MS_SV2 (.*);
```

Поскольку часто необходимо проверить, что никакие значения на кейпаде, кроме одного, не переключают состояния создадим task, который будет перебирать их:

Verilog_labs - tb_lab_MS_SV2.sv

```
19 task iterate_keypad(input int avoid_value);
20   for (int i = 0; i < 16; i++) begin
21     if (i == avoid_value) continue;
22     keypad <= i;
23     #(PERIOD);
24   end
25 endtask
```

Теперь перейдем к самому тесту. Он проходится по всем возможным ребрам графа:

Verilog_labs - tb_lab_MS_SV2.sv

```
27  initial begin
28     ENA   = 1;
29     reset = 0;
30     #PERIOD;
31     // state disarmed
32     iterate_keypad(4'b0011);
33     keypad <= 4'b0011;
34     // state armed
35     iterate_keypad(4'b1100);
36     // to disarmed
37     keypad <= 4'b1100;
38     #PERIOD;
39     // to armed
40     keypad <= 4'b0011;
41     #PERIOD;
42     // to wait_delay
43     front_door <= 1'b1;
44     #PERIOD;
45     iterate_keypad(4'b1100);
46     // to disarmed
47     front_door <= 1'b0;
48     keypad     <= 4'b1100;
49     #PERIOD;
50     // to armed
51     keypad <= 4'b0011;
52     #PERIOD;
53     // to wait_delay
54     rear_door <= 1'b1;
55     #PERIOD;
56     // to alarm
57     rear_door <= 1'b0;
58     #(100 * PERIOD);
59     iterate_keypad(4'b1100);
60     // to disarmed
61     keypad <= 4'b1100;
62     // to armed
63     keypad <= 4'b0011;
64     #PERIOD;
65     // to wait_delay
66     window <= 1'b1;
67     #PERIOD;
68     // reset to disarmed
69     reset <= 1'b1;
70     #PERIOD;
71     reset  <= 1'b0;
72     // to armed
73     keypad <= 4'b0011;
74     #PERIOD;
75     // to wait_delay
76     front_door <= 1'b1;
77     rear_door  <= 1'b1;
78     window     <= 1'b1;
79     #PERIOD;
80     $stop;
81   end
```

Теперь запустим его и посмотрим на результат:

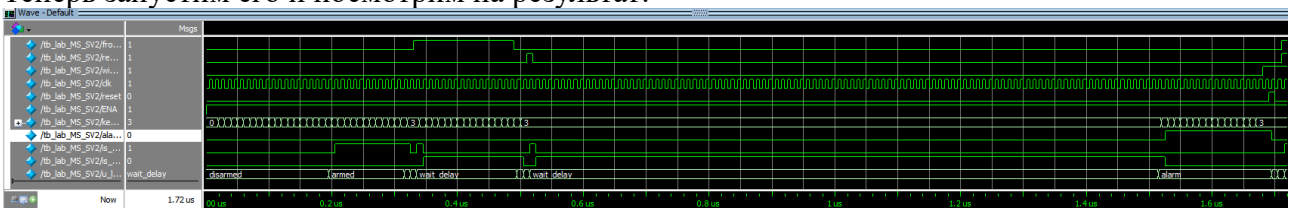

Рис. 4.3. Результат моделирования.

Как мы видим все ребра работают корректно в соответствии с ТЗ, значит пора переходить к отладке непосредственно на плате, используя ISSP и Signal Tap II.

Создадим следующий модуль для отладки:

Verilog_labs - db_lab_MS_SV2.sv

```
module db_lab_MS_SV2 (
  (* altera_attribute = "-name IO_STANDARD \"3.3-V LVCMOS\"", chip_pin = "23" *)
  input CLK
);

  bit       front_door = 0;
  bit       rear_door = 0;
  bit       window = 0;
  bit       reset = 0;
  bit       ENA = 0;
  bit [3:0] keypad = 0;
  bit       alarm_siren;
  bit       is_armed;
  bit       is_wait_delay;

  lab_MS_SV2 u_lab_MS_SV2 (
    .*,
    .clk(CLK)
  );

  SP_unit u_SP_unit (
    .source    ({front_door, rear_door, window, reset, ENA, keypad}),
    .source_clk(CLK)
  );

endmodule
```

Настроим Signal Tap II следующим образом:

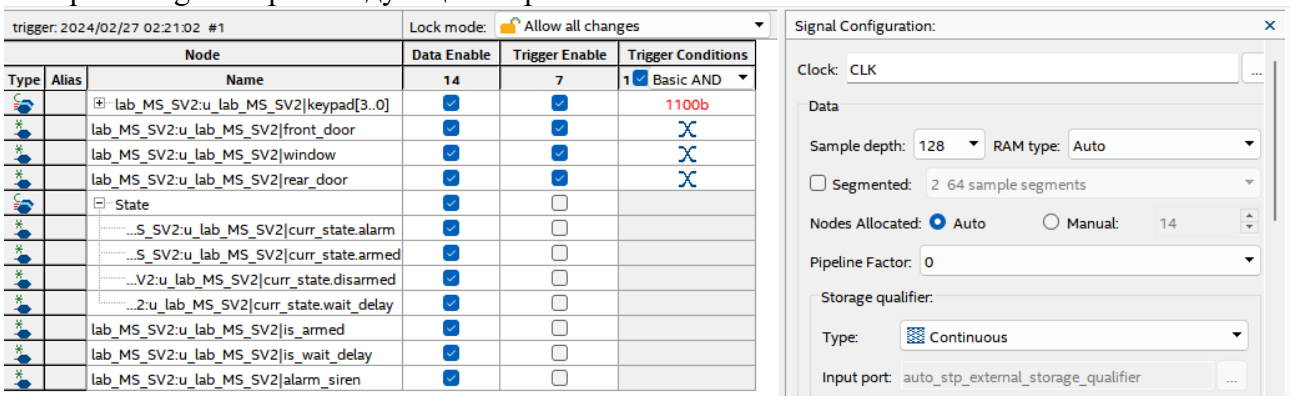

Рис. 4.4. Настройки Signal Tap II.

Также настроим ISSP:

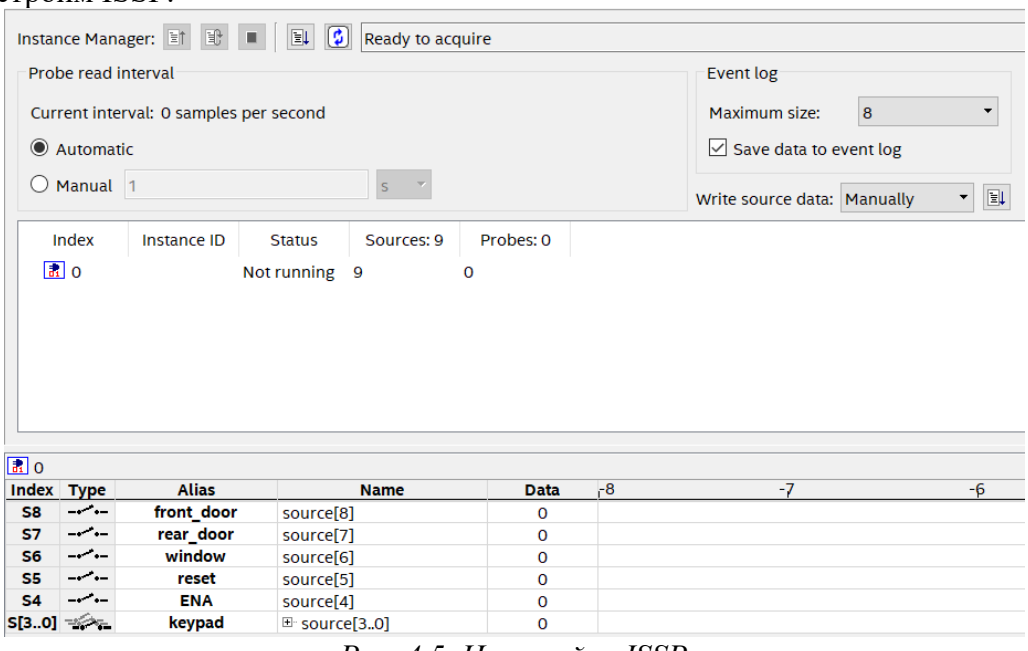

Рис. 4.5. Настройки ISSP.

Теперь запустим и выполним проверку корректности работы программы на плате. Выполним запись ENA на 1, reset на 0, keypad на 0011, таким образом перейдя в состояние armed.
Захвата не произошло. Теперь поставим keypad на 1100, перейдя в состояние disarmed, получим следующий результат в Signal Tap II:

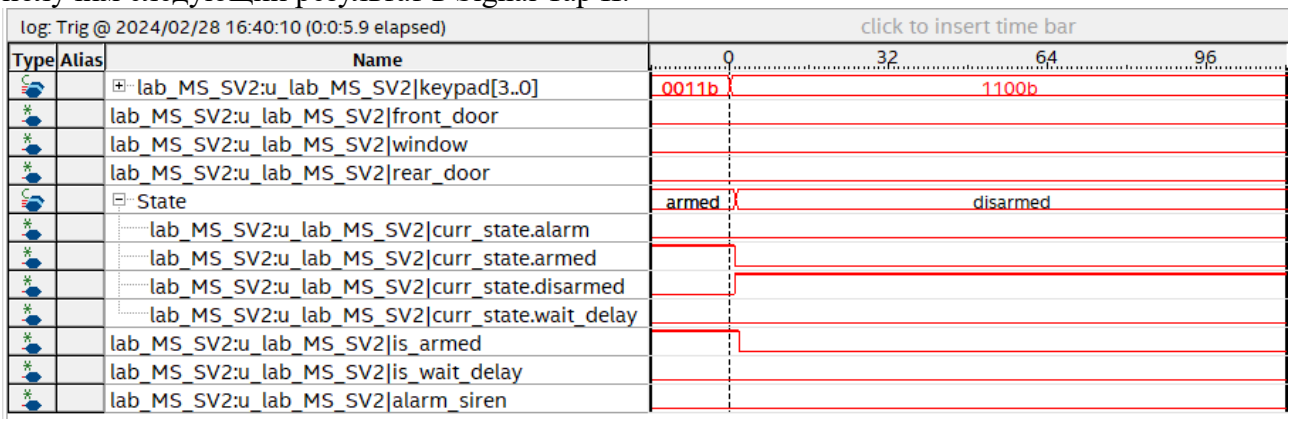

Рис. 4.6. Значения в Signal Tap II.

Теперь ставим keypad обратно на 0011, далее запишем любой из сенсоров на 1. Получим следующий результат в Signal Tap II:

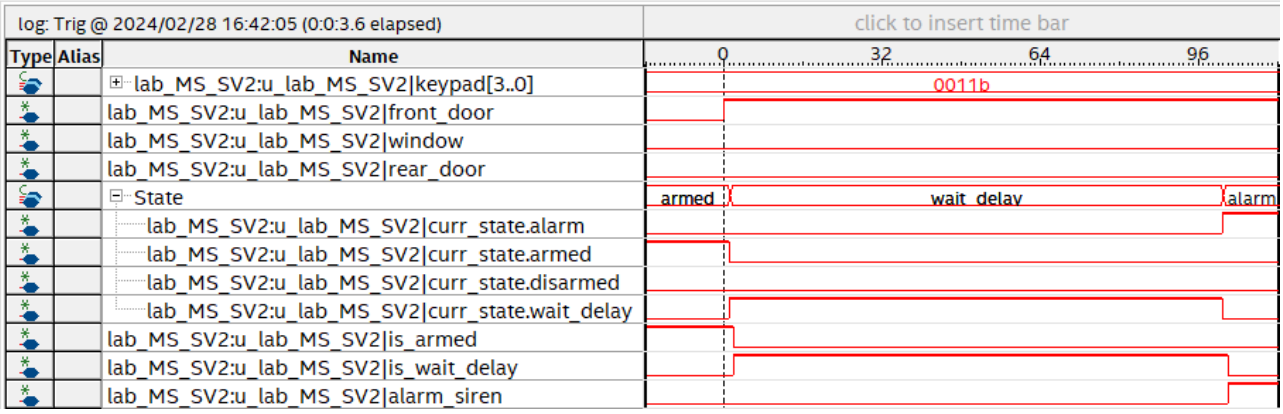

Рис. 4.7. Значения в Signal Tap II.

Произошел переход в состояние wait_delay, после чего через сто таков мы переходим в состояние alarm т.к. 100 счетов clk это очень маленькое значение.
Подадим в keypad 1100, чтоб перейти в состояние disarmed:

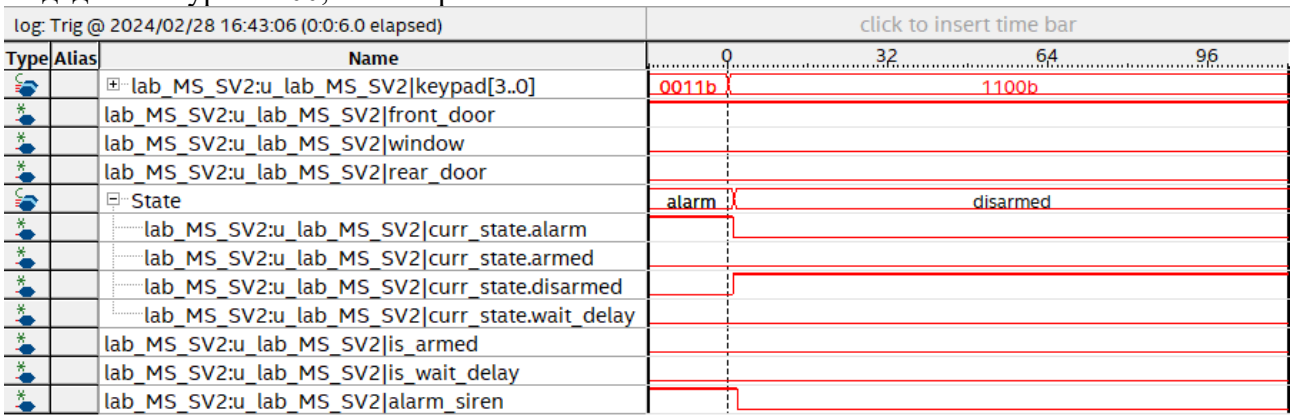

Рис. 4.8. Значения в Signal Tap II.

Как мы видим все работает корректно.
Теперь перейдем к созданию непосредственно готовой программы, для все входы перенесем на SW, выходы на светодиоды. Также сделаем счетчик делитель, который замедлит работу основного счетчика и даст нам время перед срабатыванием сигнализации:

Verilog_labs - impl_lab_MS_SV2.sv

```
`timescale 1ns / 1ns
module impl_lab_MS_SV2 (
  (* altera_attribute = "-name IO_STANDARD \"3.3-V LVCMOS\"", chip_pin = "24" *)
  input bit front_door_in,
  (* altera_attribute = "-name IO_STANDARD \"3.3-V LVCMOS\"", chip_pin = "25" *)
  input bit rear_door_in,
  (* altera_attribute = "-name IO_STANDARD \"3.3-V LVCMOS\"", chip_pin = "46" *)
  input bit window_in,
  (* altera_attribute = "-name IO_STANDARD \"3.3-V LVCMOS\"", chip_pin = "23" *)
  input bit clk,
  (* altera_attribute = "-name IO_STANDARD \"3.3-V LVCMOS\"", chip_pin = "49" *)
  input bit reset_in,
  (* altera_attribute = "-name IO_STANDARD \"3.3-V LVCMOS\"", chip_pin = "88, 89, 90, 91" *)
  input bit [3:0] keypad_in,
  (* altera_attribute = "-name IO_STANDARD \"2.5-V\"", chip_pin = "72" *)
  output bit alarm_siren,
  (* altera_attribute = "-name IO_STANDARD \"2.5-V\"", chip_pin = "71" *)
  output bit is_armed,
  (* altera_attribute = "-name IO_STANDARD \"2.5-V\"", chip_pin = "70" *)
  output bit is_wait_delay
);

  // triggers
  bit [2:0] sensors[1:0];
  bit       reset  [1:0];
  bit [3:0] keypad [1:0];

  always_ff @(posedge clk) begin
    sensors <= '{sensors[0], {front_door_in, rear_door_in, window_in}};
    keypad  <= '{keypad[0], keypad_in};
    reset   <= '{reset[0], reset_in};
  end

  // counter divider
  int i = 0;
  localparam divider = 2_500_000;

  always_ff @(posedge clk) begin
    if (i == divider) i <= 1'b0;
    else i <= i + 1'b1;
  end

  assign ENA = (i == divider);

  // module
  lab_MS_SV2 u_lab_MS_SV2 (
    .*,
    .front_door(sensors[1][2]),
    .rear_door (sensors[1][1]),
    .window    (sensors[1][0]),
    .reset     (reset[1]),
    .ENA       (ENA),
    .keypad    (keypad[1])
  );

endmodule

```

Данный модуль был залит на плату и показан преподавателю. Данная программа работала корректно в соответствии с поставленными задачами.

5. Вывод:

В ходе лабораторной работы успешно пройден цикл проектирования, начиная с создания проекта и разработки модулей с использованием расширений SystemVerilog. Использование SystemVerilog предоставило широкий спектр новых возможностей по сравнению с Verilog, облегчая процесс разработки и улучшая читаемость кода.

Отладка проекта осуществлялась с помощью инструментов In-System Sources and Probes Editor и SignalTap II, что значительно повысило эффективность процесса. Эти инструменты позволили быстро выявить и исправить ошибки, что является ключевым аспектом при работе с любым проектом.