

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

## **Отчёт по лабораторной работе Lab\_MS\_SV\_2**

Дисциплина: Автоматизация проектирования дискретных устройств (на  
английском языке)

Выполнил студент гр. 5130901/10101 \_\_\_\_\_ М.Т. Непомнящий  
(подпись)

Руководитель \_\_\_\_\_ А.А. Федотов  
(подпись)

Санкт-Петербург  
2023

## Оглавление

1. Задание.....	4
2. Ход решения.....	6
2.1. Создание модулей на языке SystemVerilog.....	6
2.2. Создание теста .....	8
2.3. Отладка устройства .....	10
2.4. Настройка SignalTap .....	12
2.5. Тестирование на плате .....	16
3. Вывод .....	17

## Список иллюстраций

Рис. 1 – Схема работы конечного автомата .....	4
Рис. 2 – Листинг кода с описанием модуля lab_MS_SV2.....	6
Рис. 3 – Листинг кода с описанием конечного автомата.....	6
Рис. 4 – Листинг кода с описанием счётчика на 100 единиц .....	7
Рис. 5 – Структура модуля lab_MS_SV2 в RTL Viewer.....	8
Рис. 6-7 – Листинг файла tb_lab_MS_SV2.sv.....	8
Рис. 8 – Симуляция проекта в ModelSim (полный вид).....	9
Рис. 9 – Переход disarmed → alarm (0–200 ns).....	9
Рис. 10 – Переход armed → disarmed → armed →wait delay (200–400 ns) .....	9
Рис. 11 – Переход wait delay → disarmed → armed →wait delay (400–600 ns).....	9
Рис. 12 – Переход wait delay → alarm → disarmed → armed (1.5–1.7 us).....	9
Рис. 13 – Листинг файла db_lab_MS_SV2.sv с кодом отладки .....	10
Рис. 14 – Структура модуля db_lab_MS_SV2 в RTL Viewer.....	10
Рис. 15 – .sdc файл с временными требованиями к проекту .....	11
Рис. 16 – Временные характеристики устройства.....	11
Рис. 17 – Настройки окна Signal Tap II.....	12
Рис. 18 – Мнемоническая таблица состояний .....	12
Рис. 19 – Окно Signal Tab II: запуск изначального состояния disarmed.....	13
Рис. 20 – Окно Signal Tab II: disarmed → armed .....	13
Рис. 21 – Окно Signal Tab II: armed → disarmed .....	13
Рис. 22 – Окно Signal Tab II: armed → wait_delay → alarm (front_door) .....	14
Рис. 23 – Окно Signal Tab II: armed → wait_delay → alarm (resr_door) .....	14
Рис. 24 – Окно Signal Tab II: armed → wait_delay → alarm (window).....	14
Рис. 25 – Окно Signal Tab II: возврат в исходное состояние disarmed .....	15
Рис. 27 – Листинг файла impl_lab_MS_SV.sv.....	16
Рис. 28 – Структура модуля impl_lab_MS_SV2 в RTL Viewer .....	17

## 1. Задание

В ходе лабораторной работы необходимо:

- 1) Разработать модуль `lab_MS_SV2` на языке SystemVerilog, добавив вход `ENA` и формирователь задержки в 100 тактов.
- 2) Создать тест `tb_lab_MS_SV2` для проверки `lab_MS_SV2` в ModelSim.
- 3) В Quartus создать модуль `SP_unit` с возможностью задания управляющих сигналов для `lab_MS_SV2` через IP модуль In-System Source and Probe.
- 4) Разработать модуль верхнего уровня `db_lab_MS_SV2`, содержащий `lab_MS_SV2` и `SP_unit`, с входом `CLK`, подключенным к тактовому сигналу на плате.
- 5) Настроить логический анализатор для `db_lab_MS_SV2` с условиями триггера `keypad[3:0] = 1100` и одновременно активными сенсорами = 1.
- 6) Провести проверку работы `db_lab_MS_SV2` на плате.
- 7) Разработать модуль `impl_lab_MS_SV2`, включающий `lab_MS_SV2` и счетчик-делитель, соединенный с `ENA`. Подключить все входы к переключателям платы и добавить по 2 последовательных регистра на каждом входе. Подключить выходы к светодиодам.

Реализовать `impl_lab_MS_SV2` на плате, проверить его работу и предоставить демонстрацию преподавателю.

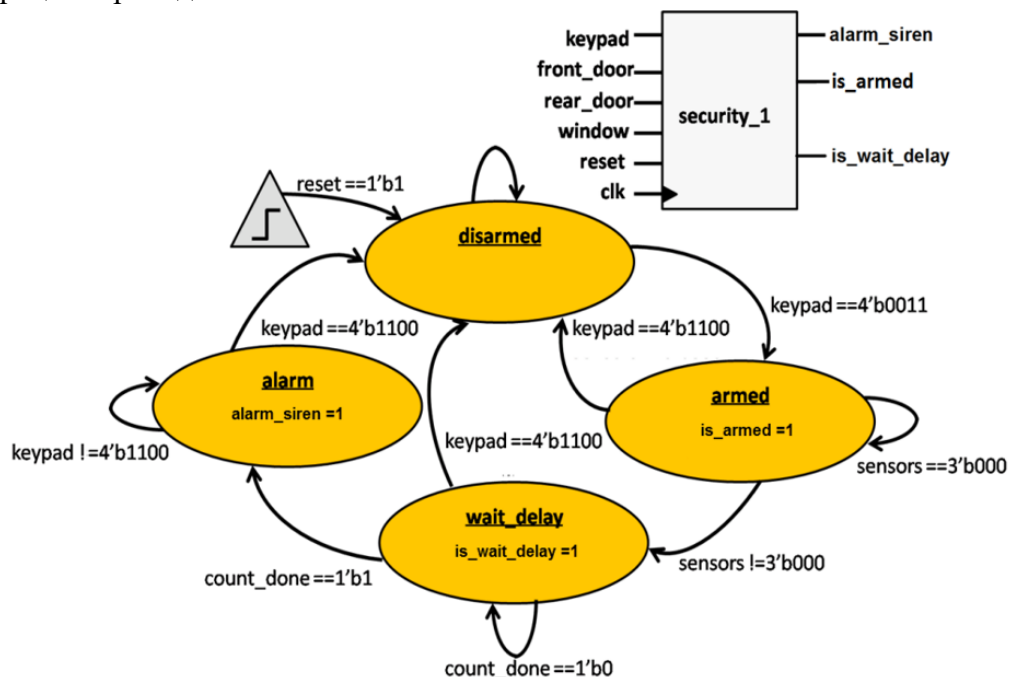


Рис. 1 – Схема работы конечного автомата

Входы:

- `CLK` – вход тактового сигнала.
- `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.)

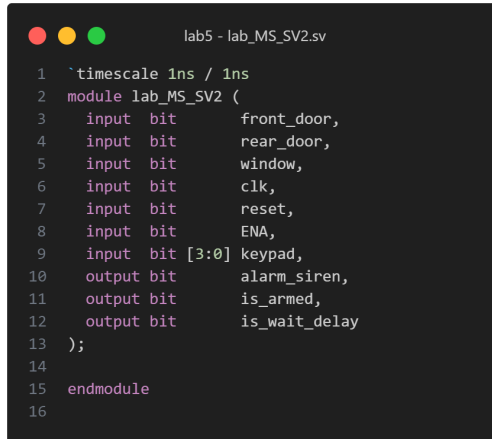
Табл. 1 – Описание переходов по рёбрам внутри графа

Ребро	Переход	Описание
keypad	disarmed -> alarm	Введен код тревоги
alarm	alarm -> armed	Система активирована
is_armed	armed -> wait_delay	Начался период задержки
count_done	wait_delay -> alarm_siren	Сирена начинает звучать
sensors	disarmed -> alarm	Сработало охранное устройство
reset	alarm -> disarmed	Система деактивирована
keypad != 4'b1100	armed -> disarmed	Неверный код
keypad == 4'b1100	disarmed -> armed	Код активации
keypad == 4'b0011	alarm -> disarmed	Код деактивации
sensors != 3'b000	disarmed -> alarm	Сработало несколько устройств
clk	all -> all	Синхронизация работы системы

## 2. Ход решения

### 2.1. Создание модулей на языке SystemVerilog

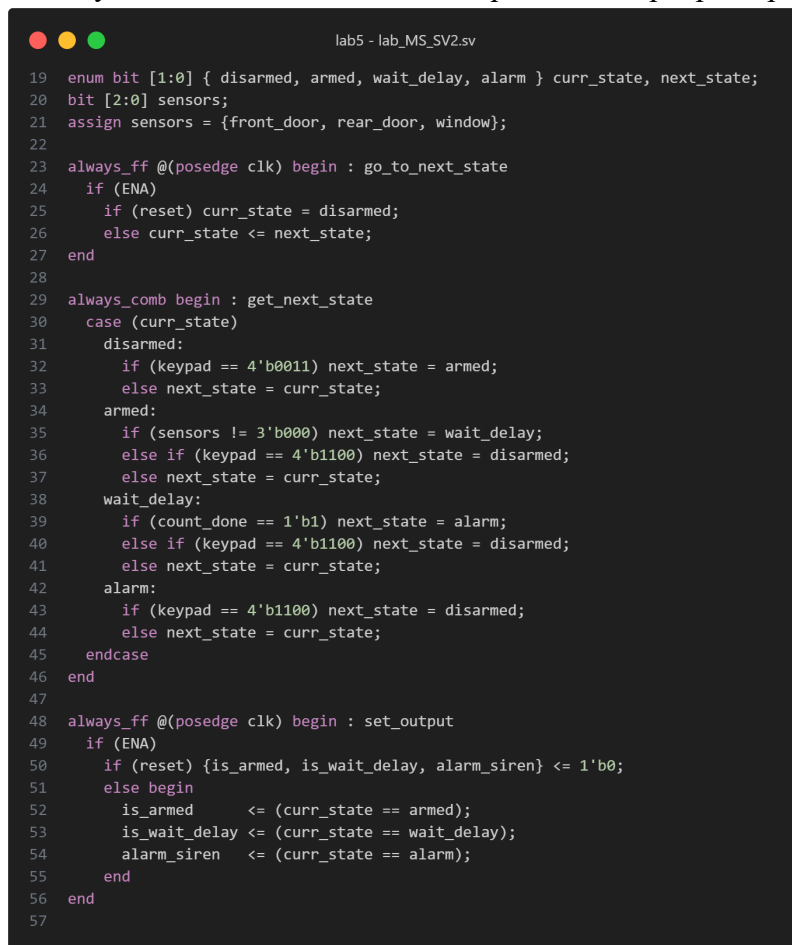
Создадим модуль lab\_MS\_SV2, задав входы и выходы типом данных bit, который позволит не думать о том, какой тип нужен net или variables (используем bit, а не logic, т. к. в этом случае получим выигрыш в скорости работы):



```
lab5 - 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
16
```

Рис. 2 – Листинг кода с описанием модуля lab\_MS\_SV2

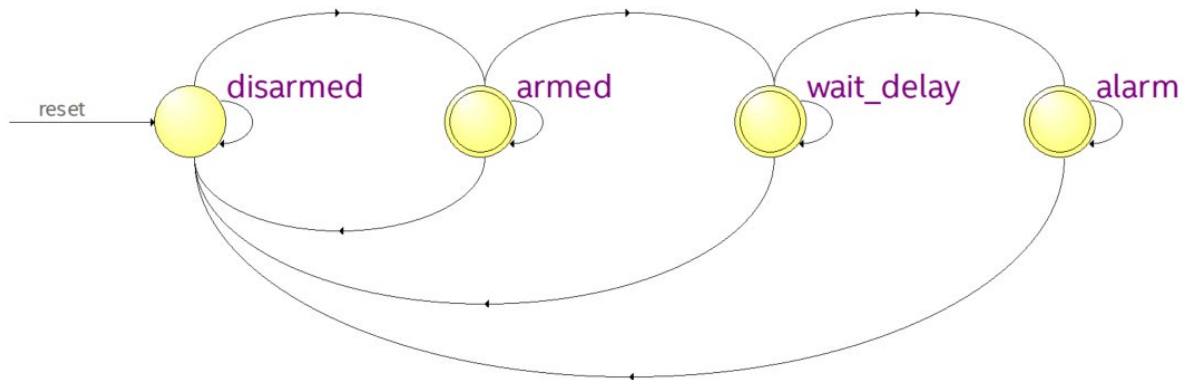
Теперь на языке SystemVerilog опишем конечный автомат, изображённый на рис. 1. Для удобства воспользуемся табл. 1 с описанием переходов по рёбрам графов:



```
lab5 - 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
57
```

Рис. 3 – Листинг кода с описанием конечного автомата

Убедиться в правильности описания конечного автомата можно воспользовавшись средствами State Machine Viewer:



Добавим описание счётчика, который будет отсчитывать время между срабатыванием сенсора и сигналом тревоги:

```

lab5 - lab_MS_SV2.sv

14  parameter delay_val = 100;
15
16  bit start_count, count_done;
17  bit [6:0] delay_cntr = 0;
18
19  ... // description of the finite state machine
20
21  assign start_count = ((curr_state == armed) && (sensors != 3'b000));
22
23  always_ff @(posedge clk) begin : counter_alarm
24      if (ENA)
25          if (reset) delay_cntr <= 0;
26          else if (start_count) delay_cntr <= delay_val - 1'b1;
27          else if (curr_state != wait_delay) delay_cntr <= 1'b0;
28          else if (delay_cntr != 0) delay_cntr <= delay_cntr - 1'b1;
29      end
30
31  assign count_done = (delay_cntr == 0);
  
```

Рис. 4 – Листинг кода с описанием счётчика на 100 единиц

Таким образом, был создан код, который представляет собой конечный автомат для системы безопасности (lab\_MS\_SV2). Модуль принимает входные сигналы, такие как состояние дверей и окон, тактовый сигнал (clk), сигнал сброса (reset), разрешение работы автомата (ENA), и ввод с клавиатуры (keypad). Код реализует логику перехода между состояниями (disarmed, armed, wait\_delay, alarm) в зависимости от текущего состояния и внешних сигналов.

В каждом такте происходит определение следующего состояния (next\_state) на основе текущего состояния (curr\_state) и входных сигналов. Выходы системы, такие как сирена (alarm\_siren), состояние вооружения (is\_armed), и ожидание задержки (is\_wait\_delay), устанавливаются в соответствии с текущим состоянием. Дополнительно, реализован счетчик задержки, который активируется в определенных условиях.

Таким образом, код обеспечивает функциональность системы безопасности, реагируя на внешние события и управляя своим состоянием в соответствии с заданным алгоритмом.

Структура данного модуля выглядит следующим образом:

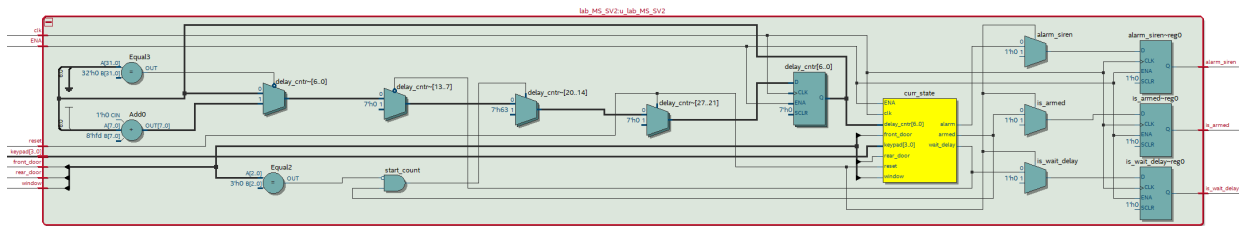


Рис. 5 – Структура модуля lab\_MS\_SV2 в RTL Viewer

## 2.2. Создание теста

На языке SystemVerilog напишем файл с тестом первого рода для созданного выше конечного автомата (рис. 3, табл. 1):

```
lab5 - 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 (.);
18
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
26
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
82
83 endmodule
```

Рис. 6-7 – Листинг файла tb\_lab\_MS\_SV2.sv

Выполним моделирование в ModelSim:



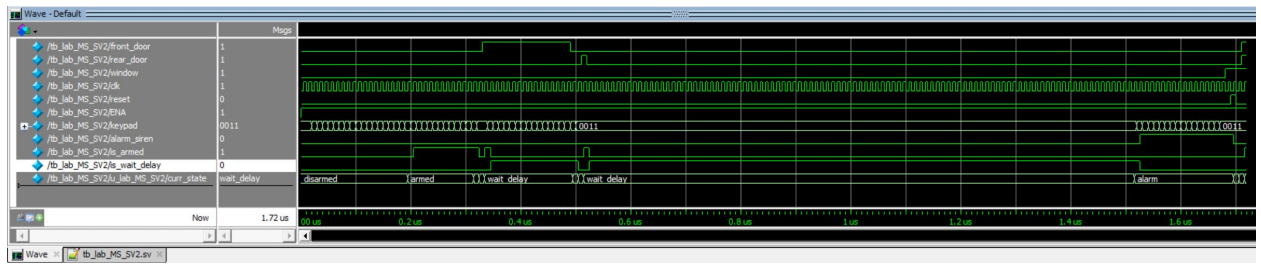


Рис. 8 – Симуляция проекта в ModelSim (полный вид)

Поскольку происходит большое количество переходов, на одном рисунке сложно показать сразу их все, поэтому приблизим те фрагменты, где происходят переходы отдельно:

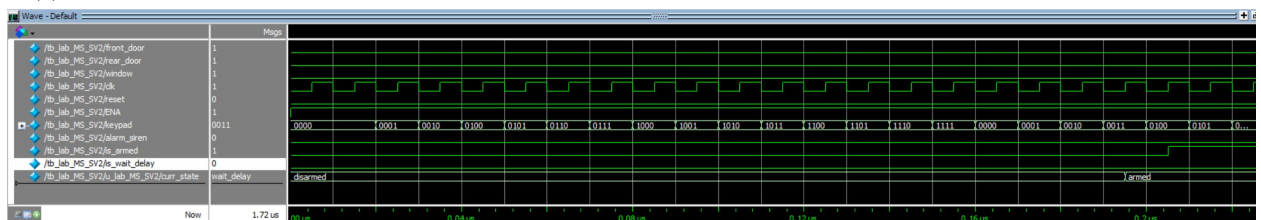


Рис. 9 – Переход disarmed → alarm (0–200 ns)

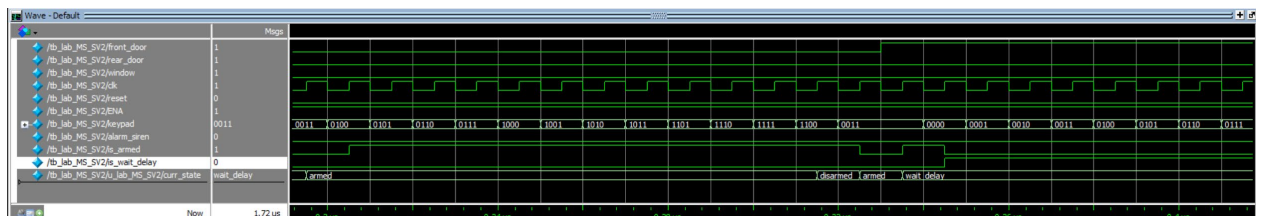


Рис. 10 – Переход armed → disarmed → armed → wait delay (200–400 ns)

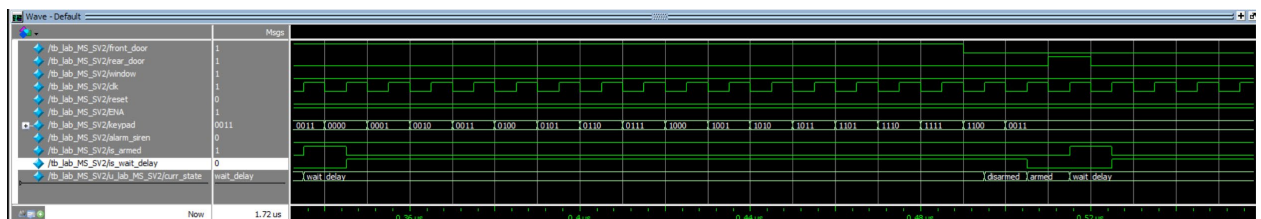


Рис. 11 – Переход wait delay → disarmed → armed → wait delay (400–600 ns)

Далее, до примерно 1.5 us никаких изменений не происходит, поэтому рассмотрим сразу следующий переход:

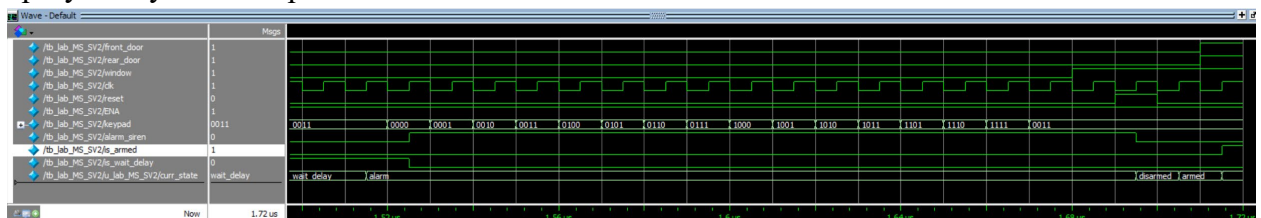


Рис. 12 – Переход wait delay → alarm → disarmed → armed (1.5–1.7 us)

Из приведённых выше скриншотов временной диаграммы видно, что тестовый модуль успешно проходит через все состояния конечного автомата, описанных в задании.

## 2.3. Отладка устройства

Для отладки устройства создадим и настроим ISSPE модуль. Также необходимо создать файл верхнего уровня для отладки. Структура модуля для отладки приведена на рис. 12 ниже:

```
lab5 - db_lab_MS_SV2.sv

1 module db_lab_MS_SV2 (
2   (* altera_attribute = "-name IO_STANDARD \"3.3-V LVC MOS\"", chip_pin = "23" *)
3   input CLK
4 );
5
6 bit    front_door = 0;
7 bit    rear_door = 0;
8 bit    window = 0;
9 bit    reset = 0;
10 bit    ENA = 0;
11 bit [3:0] keypad = 0;
12 bit    alarm_siren;
13 bit    is_armed;
14 bit    is_wait_delay;
15
16 lab_MS_SV2 u_lab_MS_SV2 (
17   .*,
18   .clk(CLK)
19 );
20
21 SP_unit u_SP_unit (
22   .source ({front_door, rear_door, window, reset, ENA, keypad}),
23   .source_clk(CLK)
24 );
25
26 endmodule
27
```

Рис. 13 – Листинг файла db\_lab\_MS\_SV2.sv с кодом отладки

Его структура в RTL Viewer выглядит следующим образом:

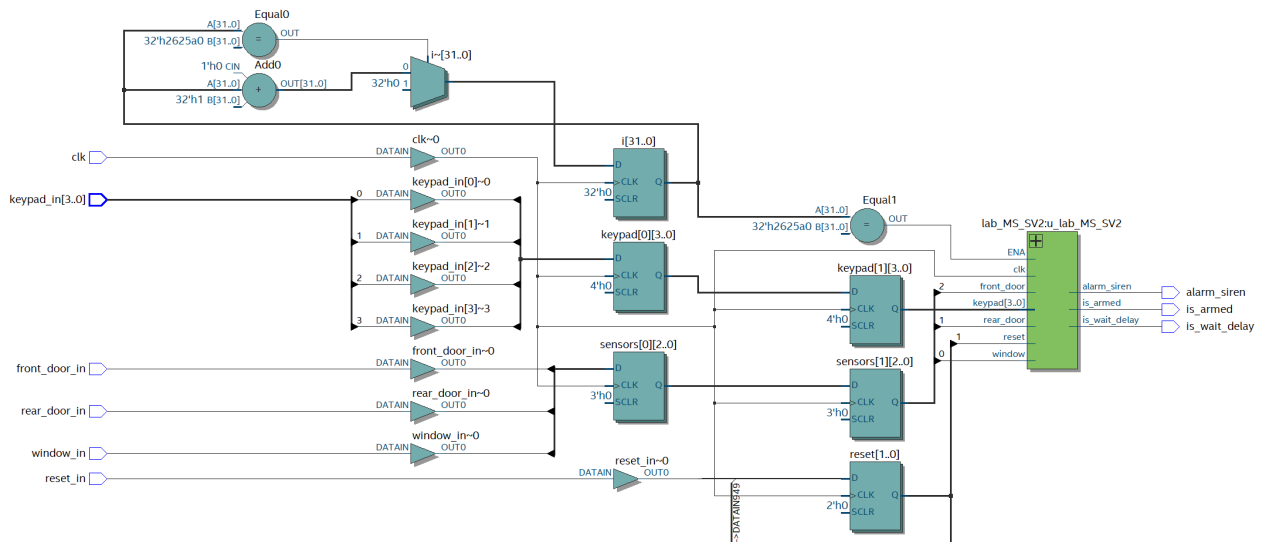


Рис. 14 – Структура модуля db\_lab\_MS\_SV2 в RTL Viewer

Создадим .sdc файл с требованиями к тактовому сигналу:

```
1 create_clock -name {clk} -period 40.000 [get_ports {clk}]
2 derive_clock_uncertainty
```

Рис. 15 – .sdc файл с временными требованиями к проекту

После компиляции посмотрим на временные характеристики модели:

Slow 1200mV 85C Model Fmax Summary				
<<Filter>>				
	Fmax	Restricted Fmax	Clock Name	
1	72.08 MHz	72.08 MHz	altera...ed_tck	
2	145.84 MHz	145.84 MHz	clk	

Рис. 16 – Временные характеристики устройства

## 2.4. Настройка SignalTap

Для отображения выводов конечного автомата создадим мнемоническую. Зададим следующие настройки Signal Tap II:

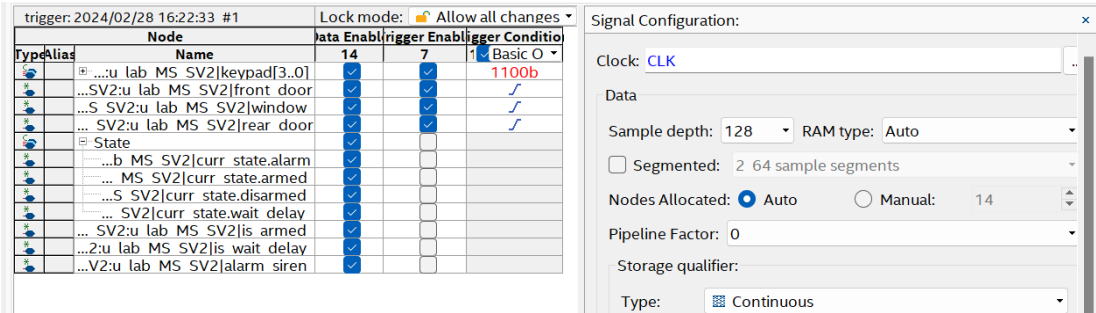


Рис. 17 – Настройки окна Signal Tap II

Выполним полную компиляцию. В отчете о компиляции видно, что устройство удовлетворяет временным параметрам.

Slow 1200mV 85C Model Fmax Summary				
<<Filter>>				
	Fmax	Restricted Fmax	Clock Name	N
1	50.02 MHz	50.02 MHz	altera...ed_tck	
2	142.82 MHz	142.82 MHz	CLK	

Рис. 15 – Временные характеристики устройства

Для удобного отображения значения состояний в окне Signal Tap II создадим мнемоническую таблицу, которая будет выводить состояния словами при подаче соответствующего сигнала в Signal Probe:

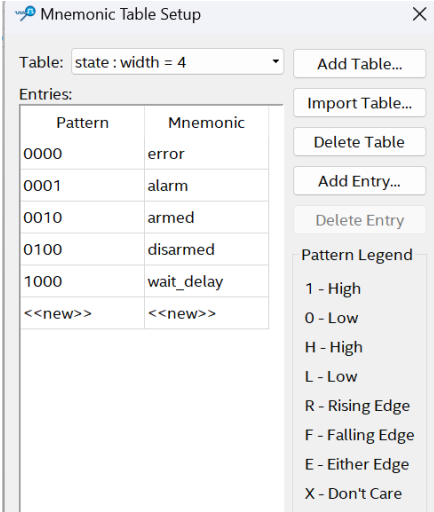


Рис. 18 – Мнемоническая таблица состояний

Теперь запустим и выполним проверку корректности работы программы на плате. Изначально мы находимся в состоянии disarmed:

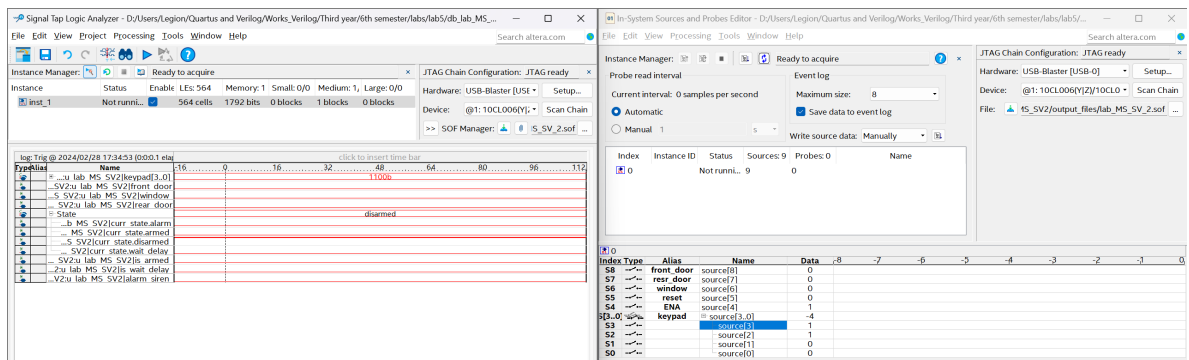


Рис. 19 – Окно Signal Tab II: запуск изначального состояния disarmed

Выполним запись ENA на 1, reset на 0, keypad на 0011, таким образом перейдя в состояние armed. Захвата не произошло. Однако, при перезаписи захвата сигнала можно будет увидеть, что в данный момент мы находимся в состоянии armed:

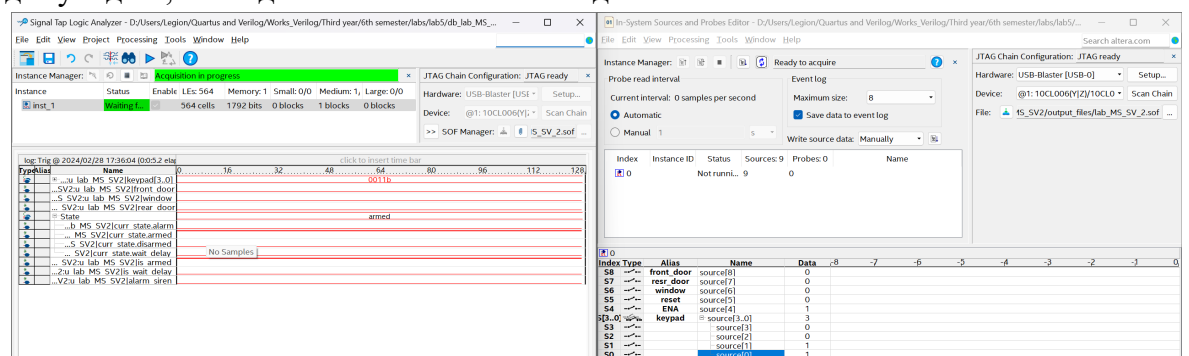


Рис. 20 – Окно Signal Tab II: disarmed → armed

Теперь поставим keypad на 1100, перейдя в состояние disarmed, получим следующий результат в Signal Tab II:

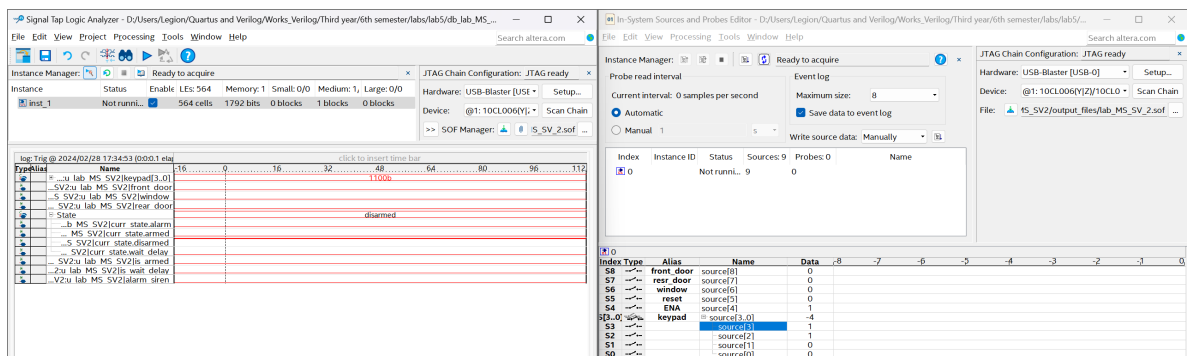


Рис. 21 – Окно Signal Tab II: armed → disarmed

Снова перейдём в состояние armed, подав на keypad 0011. Протестируем работу сигналов front\_door, resr\_door, window, проверяя, что система безопасности обрабатывает ситуацию, когда кто-то пытается забраться в окно или одну из дверей:

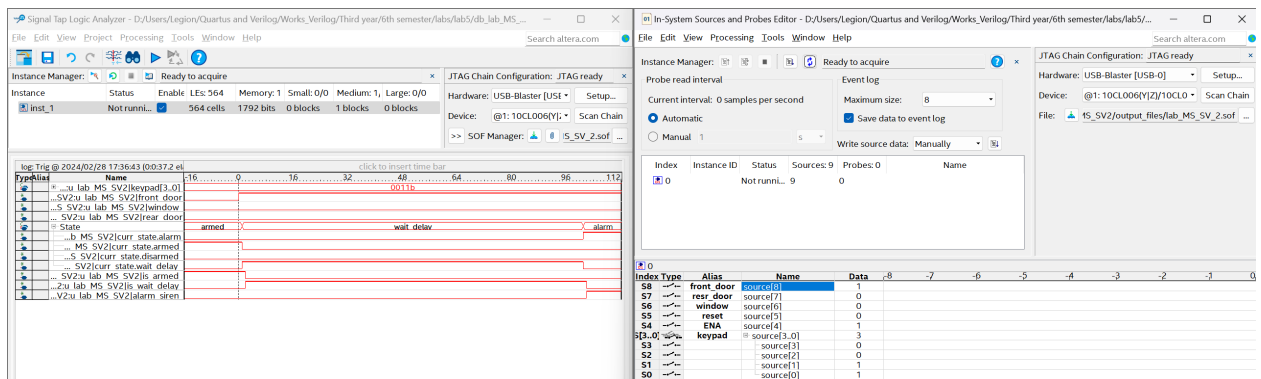


Рис. 22 – Окно Signal Tab II: armed → wait\_delay → alarm (front\_door)

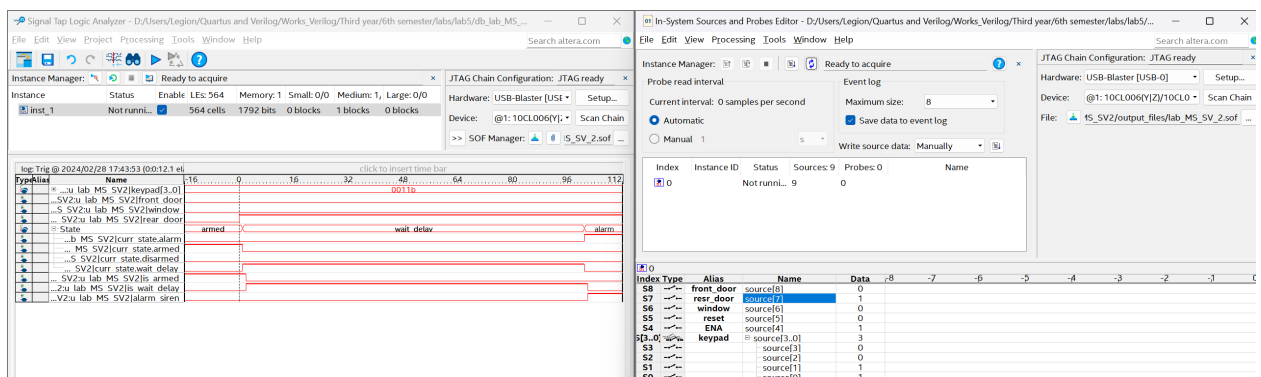


Рис. 23 – Окно Signal Tab II: armed → wait\_delay → alarm (resr\_door)

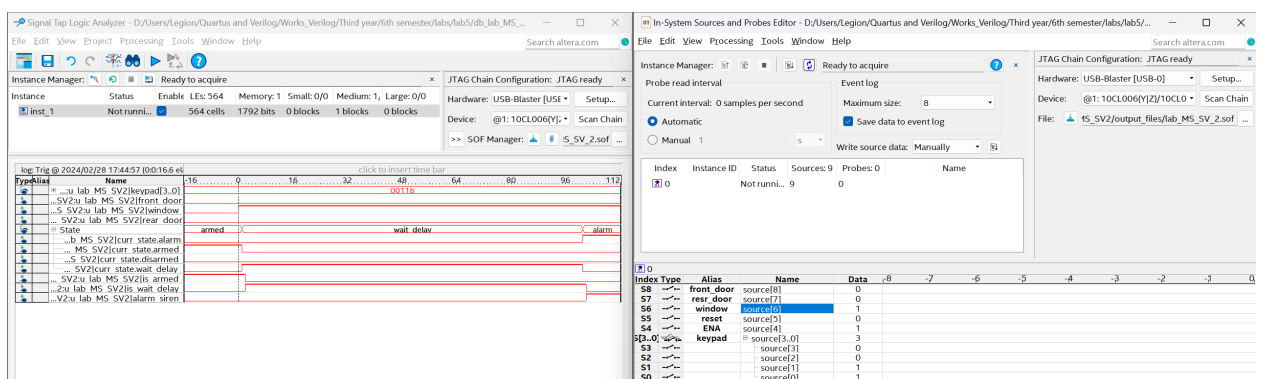


Рис. 24 – Окно Signal Tab II: armed → wait\_delay → alarm (window)

Заметим, что произошел переход в состояние wait\_delay, после чего почти сразу мы переходим в состояние alarm т. к.  $clk = 100$ , что является очень маленьким значением.

\*При реализации блока для демонстрации на плате будет использован счётчик-делитель, который будет реализовывать те самые 10 секунд, которые проходя перед тем, как система перейдёт из wait\_delay в alarm:

Подадим на keypad значение 1100, тем самым перейдя в исходное состояние disarmed:

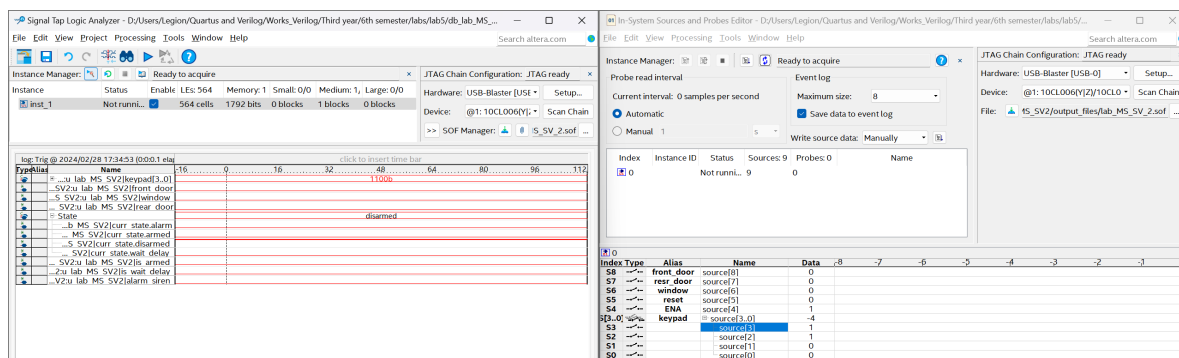


Рис. 25 – Окно Signal Tab II: возврат в исходное состояние disarmed

Таким образом, были протестированы все возможные переходы, и работа системы безопасности полностью соответствует ожиданиям.

## 2.5. Тестирование на плате

Перейдём к созданию программного модуля `impl_lab_MS_SV.sv` с последующим его тестированием на плате. Переход по состояниям, а также сброс происходят при помощи SW. За всеми изменениями можно следить с помощью светодиодов, на которые выведены соответствующие выходы.

```
lab5 - impl_lab_MS_SV2.sv

1  `timescale 1ns / 1ns
2  module impl_lab_MS_SV2 (
3      (* altera_attribute = "-name IO_STANDARD \"3.3-V LVCMS\"", chip_pin = "24" *)
4      input bit front_door_in,
5      (* altera_attribute = "-name IO_STANDARD \"3.3-V LVCMS\"", chip_pin = "25" *)
6      input bit rear_door_in,
7      (* altera_attribute = "-name IO_STANDARD \"3.3-V LVCMS\"", chip_pin = "46" *)
8      input bit window_in,
9      (* altera_attribute = "-name IO_STANDARD \"3.3-V LVCMS\"", chip_pin = "23" *)
10     input bit clk,
11     (* altera_attribute = "-name IO_STANDARD \"3.3-V LVCMS\"", chip_pin = "88" *)
12     input bit reset_in,
13     (* altera_attribute = "-name IO_STANDARD \"3.3-V LVCMS\"", chip_pin = "49, 91, 90, 89" *)
14     input bit [3:0] keypad_in,
15     (* altera_attribute = "-name IO_STANDARD \"2.5-V\"", chip_pin = "72" *)
16     output bit alarm_siren,
17     (* altera_attribute = "-name IO_STANDARD \"2.5-V\"", chip_pin = "71" *)
18     output bit is_armed,
19     (* altera_attribute = "-name IO_STANDARD \"2.5-V\"", chip_pin = "70" *)
20     output bit is_wait_delay
21 );
22
23 // triggers
24 bit [2:0] sensors[1:0];
25 bit      reset  [1:0];
26 bit [3:0] keypad [1:0];
27
28 always_ff @(posedge clk) begin
29     sensors = '{sensors[0], {front_door_in, rear_door_in, window_in}};
30     keypad  = '{keypad[0], keypad_in};
31     reset   = '{reset[0], reset_in};
32 end
33
34 // counter divider
35 int i = 0;
36 localparam divider = 2_500_000;
37
38 always_ff @(posedge clk) begin
39     if (i == divider) i <= 1'b0;
40     else i <= i + 1'b1;
41 end
42
43 assign ENA = (i == divider);
44
45 // module
46 lab_MS_SV2 u_lab_MS_SV2 (
47     .*,
48     .front_door(sensors[1][2]),
49     .rear_door (sensors[1][1]),
50     .window    (sensors[1][0]),
51     .reset     (reset[1]),
52     .ENA       (ENA),
53     .keypad    (keypad[1])
54 );
55
56 endmodule
57
```

Рис. 26 – Листинг файла `impl_lab_MS_SV.sv`

Его структура в RTL Viewer выглядит следующим образом:



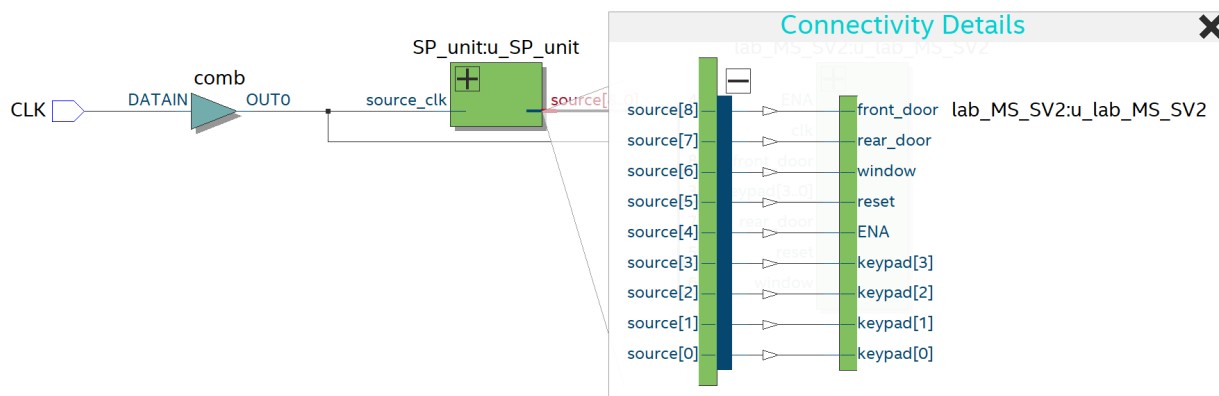


Рис. 27 – Структура модуля impl\_lab\_MS\_SV2 в RTL Viewer

На рис. 28 также представлено окно Connectivity Details, которое показывает, что все элементы правильно подключены друг к другу (проверка того, что при тестировании полученные состояния отображаются правильно).

Средствами Programmer QP разработанный модуль был загружен на плату. Были протестированы переходы по всем возможным состояниям конечного автомата (табл. 1). Как и ожидалось, все переходы осуществлялись корректно в соответствии с заданием.

### 3. Вывод

В результате выполнения лабораторной работы успешно реализован конечный автомат системы безопасности lab\_MS\_SV2 (рис. 3) с использованием расширений SystemVerilog. Создан тестовый модуль tb\_lab\_MS\_SV2 (рис. 5), который успешно проверил корректность переходов и формирование задержки. Разработан модуль SP\_unit в Quartus для управления lab\_MS\_SV2 без кнопок на плате. Модуль верхнего уровня db\_lab\_MS\_SV2, включающий lab\_MS\_SV2 и SP\_unit, успешно настроен для исследования и отладки с использованием логического анализатора. Реализованный модуль impl\_lab\_MS\_SV2 (рис. 12) включает счетчик-делитель, обеспечивающий нужную частоту, и успешно интегрирован на плату, что было проверено и продемонстрировано преподавателю. Теоретические результаты работы программного кода, описывающего конечный автомат, были получены средствами ModelSim и совпали с теми, которые были получены при подключении самой платы.

Отметим, что переход с Verilog на SystemVerilog был эффективным, предоставив более удобное написание кода за счёт использования более ёмких/кратких структур набора команд. Это улучшило структурирование кода и позволило эффективно работать над созданием конечного автомата.