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

Отчёт по лабораторной работе № 7

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

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

Руководитель _____ А.А. Федотов
(подпись)

“23” марта 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:

Структура разрабатываемого устройства приведена на рисунке ниже.

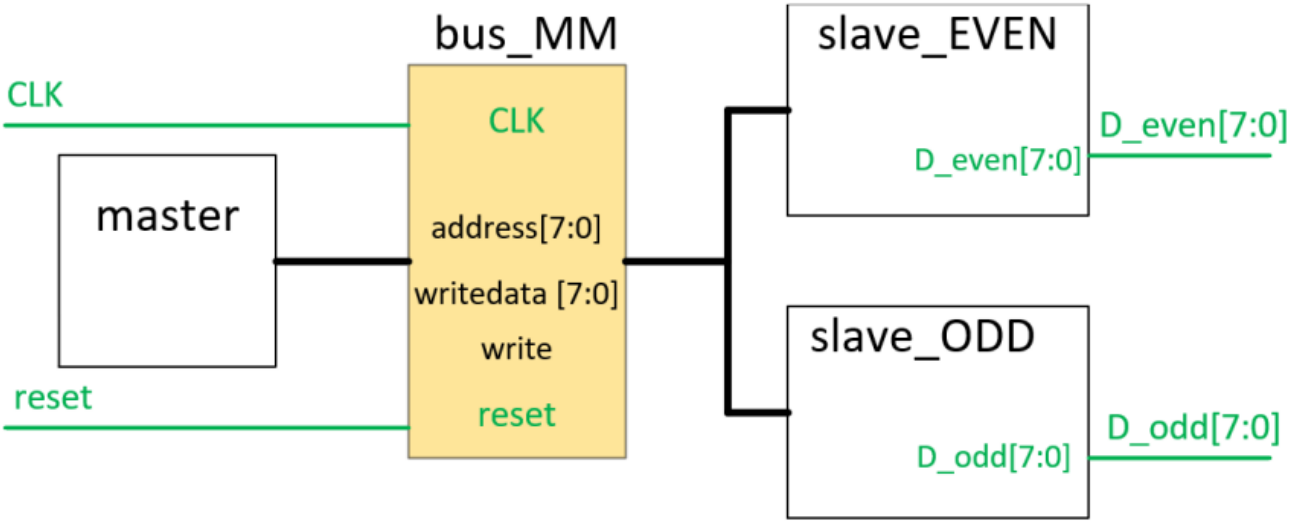

Рис. 1. Структура разрабатываемого устройства.

В состав устройства входят:

- Модуль master – ведущее устройство, формирует обращение к двум ведомым устройствам.
- Модули slave_EVEN и slave_ODD – ведомые устройства, управляемые мастером.
- Модуль bus_MM - экземпляр интерфейса, обеспечивающий подключение мастера и ведомых устройств.

Выводы устройства (выделены зеленым цветом):

- CLK – вход тактового сигнала.
- reset – синхронный сброс всех устройств.
- D_odd – восьмиразрядный выход.
- D_even – восьмиразрядный выход.

Алгоритм работы разрабатываемого устройства определяется алгоритмами работы его модулей:

- Модуль master:
  - содержит КА Мура:
    - 3 состояния (граф переходов приведен ниже) - все переходы между состояниями безусловные:
      - начальное – initSM.
      - пустое – nop.
      - записи данных -wr1D.
    - формирует сигналы
      - address - Адреса (8 бит); комбинационный выход.
      - writedata - Данных (8 бит); комбинационный выход;
      - write - Разрешения записи; комбинационный выход;
  - содержит счетчик cnt (счетчик 8-ми разрядный), его значение используется для формирования адреса (address) и данных (writedata).

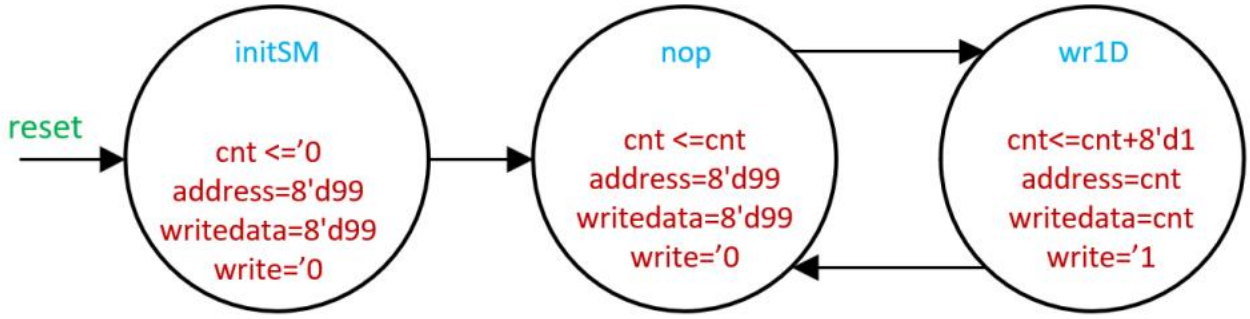

Рис. 2. Переходы конечного автомата.

- Модуль slave_EVEN - «ведомый_четный»:
  - анализирует address и сигнал write и если: address четный и write = 1, то записывает в свой внутренний регистр значение шины данных (writedata);
  - на выходе – значение внутреннего регистра;
- Модуль slave_ODD - «ведомый_нечетный»:
  - анализирует address и сигнал write и если: address нечетный и write = 1, то записывает в свой внутренний регистр значение шины данных (writedata);
  - на выходе – значение внутреннего регистра;

4. Решение:

Выполним описание интерфейса, для разрабатываемых модулей:

```
interface bus_MM (
  input bit CLK,
  input bit reset
);
  bit [7:0] address;
  bit [7:0] writedata;
  bit       write;
endinterface
```

Используя интерфейс, создадим описание «мастера»:

```
`timescale 1ns / 1ns
module master (
  bus_MM bus
);
  enum bit [1:0] {initSM, nop, wr1D}       fsm_MM;
  bit                              [7:0] cnt;

  always_ff @(posedge bus.CLK) begin
    if (bus.reset) begin
      fsm_MM <= initSM;
      cnt    <= '0;
    end else
      case (fsm_MM)
        initSM: fsm_MM <= nop;
        nop:    fsm_MM <= wr1D;
        wr1D: begin
          fsm_MM <= nop;
          cnt    <= cnt + 8'd1;
        end
      endcase
  end

  always_comb begin
    case (fsm_MM)
      wr1D: begin
        bus.address   = cnt;
        bus.write     = '1;
        bus.writedata = cnt;
      end
      default: begin
        bus.address   = 8'd99;
        bus.write     = 1'd0;
        bus.writedata = 8'd99;
      end
    endcase
  end

endmodule
```

В данном модуле реализован конечный автомат Мура в соответствии с заданием.
Теперь разработаем модули slave_ODD и slave_EVEN, используя разработанный интерфейс:

```
`timescale 1ns / 1ns
module slave_EVEN (
         bus_MM       bus,
  output bit    [7:0] D_even
);

  always_ff @(posedge bus.CLK)
    if (bus.reset) D_even <= '0;
    else if ((bus.address ==? 8'b???????0) & (bus.write == '1)) D_even <= bus.writedata;

endmodule
```

```
`timescale 1ns / 1ns
module slave_ODD (
         bus_MM       bus,
  output bit    [7:0] D_odd
);

  always_ff @(posedge bus.CLK)
    if (bus.reset) D_odd <= '0;
    else if ((bus.address ==? 8'b???????1) & (bus.write == '1)) D_odd <= bus.writedata;

endmodule
```

Как мы видим запись в EVEN происходит по сигналу write и адресу, оканчивающемуся на 0, а в ODD, наоборот, адрес должен кончаться на 1.
Теперь разработаем описание верхнего уровня:

```
`timescale 1ns / 1ns
module lab_MS_SV5 (
  input  bit       CLK,
  input  bit       reset,
  output bit [7:0] D_even,
  output bit [7:0] D_odd
);

  bus_MM bus (.*);
  master UUT_master (.*);
  slave_EVEN UUT_slave_EVEN (.*);
  slave_ODD UUT_slave_ODD (.*);

endmodule
```

RTL Viewer разработанной схемы выглядит следующим образом:

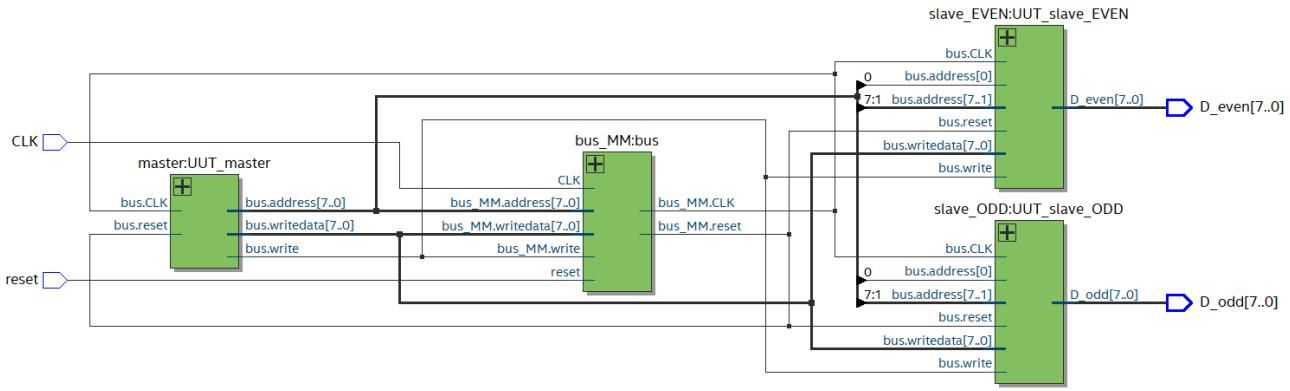

Рис. 3. RTL Viewer устройства.

Разработаем тест первого класса для устройства:

```
1  `timescale 1ns / 1ns
2  module tb_lab_MS_SV5 ();
3    bit       CLK;
4    bit       reset = '1;
5    bit [7:0] D_even;
6    bit [7:0] D_odd;
7    lab_MS_SV5 UUT (.*);
8    initial forever #5 CLK = ~CLK;
9
10   initial begin
11     #7;
12     reset = '0;
13     repeat (32) @(negedge CLK);
14     $stop;
15   end
16 endmodule
```

Запустим тест и посмотрим, корректно ли работает разработанное устройство:

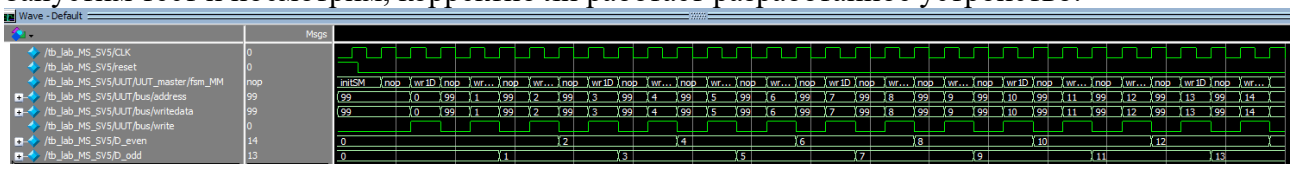

Рис. 4. Результат тестирования.

Как мы можем заметить, все работает корректно.

Теперь разработаем модуль для отладки устройства на плате:

```
1  module db_lab_MS_SV5 (
2    (* altera_attribute = "-name IO_STANDARD \"3.3-V LVCMOS\"", chip_pin = "23" *)
3    input bit CLK
4  );
5
6    bit       reset;
7    bit [7:0] D_even;
8    bit [7:0] D_odd;
9    lab_MS_SV5 UUT (.*);
10   SP_unit SU_ (
11     .source    (reset),
12     .source_clk(CLK)
13   );
14
15 endmodule
```

Используя SP, будем подавать reset, а данные снимать будем с использованием SignalTap II:

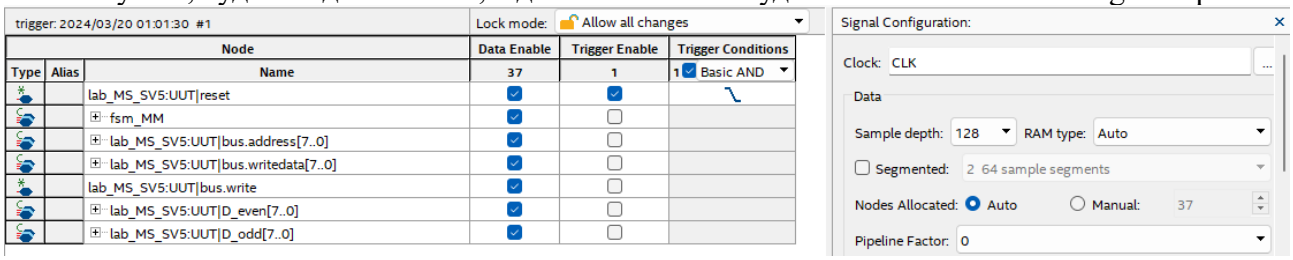

Рис. 5. Настройки Signal Tap II.

Запишем модуль на плату, включим Signal Tap II и переключим reset в 0. Тогда получим следующий результат:

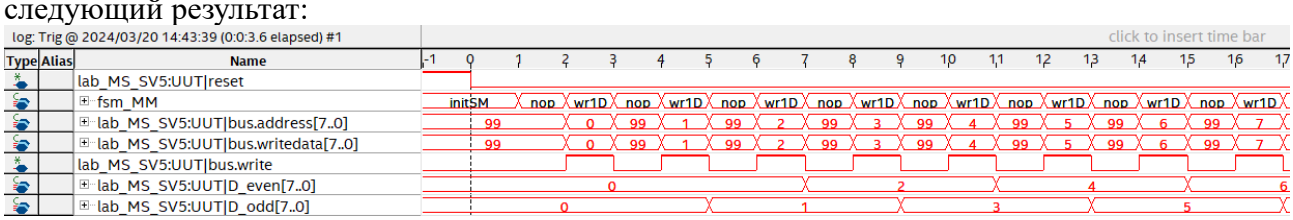

Рис. 6. Signal Tap II после запуска.

Как мы видим, после установки reset в 0 мы перешли из стартового состояния в цикл из состояний nop и wr1D. D_EVEN и D_ODD меняются в соответствии с заданием, что свидетельствует о корректности разработанного устройства.
Теперь немного изменим интерфейс:

```
interface bus_MM (
  input bit CLK,
  input bit reset
);
  bit [7:0] address;
  bit [7:0] writedata;
  bit       write;

  modport master(input CLK, reset, output address, writedata, write);
  modport slave(input CLK, reset, address, writedata, write);
endinterface
```

Вместо того, чтоб оставлять входы и выходы inout явно зададим что есть что. Из-за этого чуть-чуть изменится объявления интерфейсов в различных модулях:

```
module master (
  bus_MM.master bus
);
```

```
module slave_EVEN (
        bus_MM.slave       bus,
  output bit          [7:0] D_even
);
```

```
module slave_ODD (
        bus_MM.slave       bus,
  output bit          [7:0] D_odd
);
```

Более ничего не поменялось в коде. Выполним компиляцию и посмотрим на RTL Viewer:

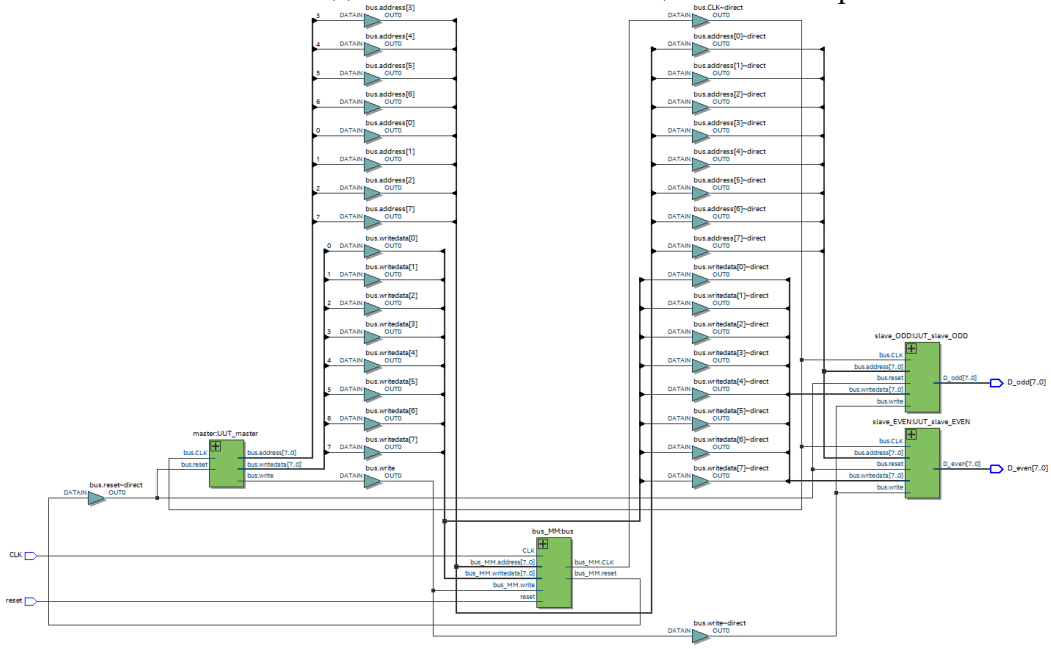

Рис. 7. RTL Viewer модуля с измененным интерфейсом.

Как мы видим, появились дополнительные элементы, они позволяют настроить направление данных, то, что мы и задавали в интерфейсе.
Теперь чуть изменим настройки для SignalTap II:

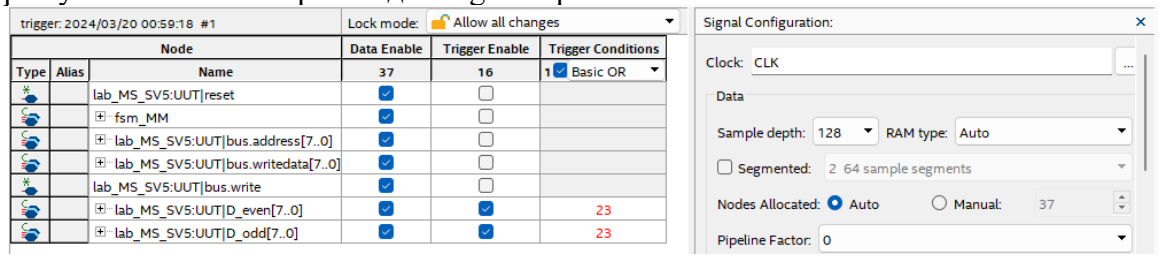

Рис. 8. Signal Tap II. Обновленные настройки.

Запишем на плату разработанный ранее отладочный модуль и посмотрим на результат в Signal Tap II:

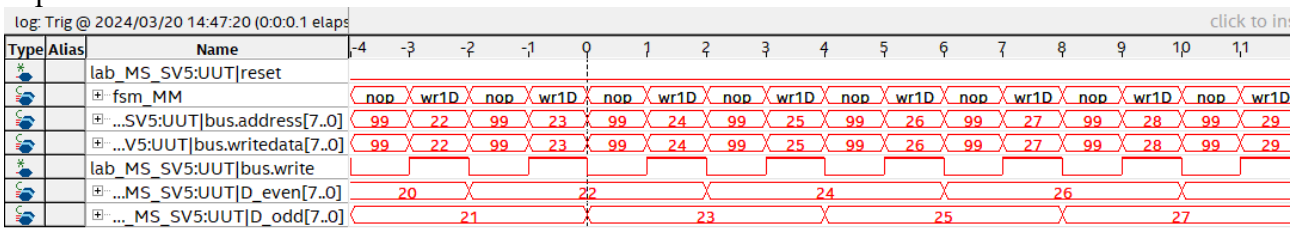

Рис. 9. Signal Tap II.

Как мы видим, мы зафиксировали значение 23, что и требовалось по заданию.

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.