

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

Отчёт по лабораторной работе Lab_PD4

Дисциплина: Автоматизация проектирования дискретных устройств (на
английском языке)

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

Руководитель _____ А.А. Федотов
(подпись)

Санкт-Петербург
2024

Оглавление

1.	Задание.....	5
2.	Ход работы	6
2.1.	Создание описания модулей.....	6
	Создание модуля my_master	6
	Создание модуля my_slave.....	9
	Создание модуля my_Dslave	10
	Отличие my_slave от my_Dslave.....	10
	Принцип работы системы	10
2.2.	Создание проекта.....	11
	Начало работы в PD	11
2.3.	Настройка сигналов.....	14
	Настройка clk.....	14
2.4.	Подключение сигналов	14
	Подключение тактового сигнала	14
	Подключение сигнала Reset.....	15
2.5.	Подключение Avalon-MM интерфейсов	15
2.6.	Экспорт выводов.....	17
2.7.	Анализ системы	17
	Проверка блока.....	17
	Анализ с помощью Schematic	19
	Генерация системы	19
2.8.	Подключение файлов к проекту.....	20
3.	Тестирование проекта	21
3.1.	Тестирование средствами ModelSim	21
	Создание тестового файла.....	21
	Симуляция средствами ModelSim	22
	Изменение тестового файла	23
	Симуляция средствами ModelSim (все значения).....	23
3.2.	Тестирование средствами Signal Tap II.....	24
	Создание файла для отладки.....	24
	Настройка Signal Tap II.....	25
	Тестирование на плате средствами Signal Tap II	25
4.	Вывод.....	27

Список иллюстраций

Рис. 1 – Структура проекта.....	5
Рис. 2 – Модуль my_master.....	6
Рис. 3 – Схема конечного автомата	7
Рис. 4 – Модуль my_master в RTL Viewer.....	8
Рис. 5 – Схема конечного автомата в State Machine Viewer.....	8
Рис. 6 – Модуль my_slave	9
Рис. 7 – Модуль my_slave в RTL Viewer	9
Рис. 8 – Модуль my_Dslave	10
Рис. 9 – Модуль my_slave в RTL Viewer	10
Рис. 10 – Исходное окно PD	11
Рис. 11 – Зкладка Component Type в New Component.....	11
Рис. 12 – Зкладка Files в New Component.....	11
Рис. 13 – Выбор файла для симуляции.....	12
Рис. 14 – Зкладка Signals & Interfaces в New Component (master).....	12
Рис. 15 – Проверка успешного добавления мастера в проект.....	12
Рис. 16 – Зкладка Signals & Interfaces в New Component (slave)	13
Рис. 17 – Зкладка Signals & Interfaces в New Component (Dslave)	13
Рис. 18 – Проверка успешности добавления модулей	13
Рис. 19 – Добавление модулей в систему.....	14
Рис. 20 – Настройка компонента clk.....	14
Рис. 21 – Подключение тактового сигнала (1).....	15
Рис. 22 – Подключение тактового сигнала (2).....	15
Рис. 23 – Подключение сигнала Reset	15
Рис. 24 – Подключение Avalon-MM интерфейсов	16
Рис. 25 – Фиксация адресов	16
Рис. 26 – Назначение правильных адресов для компонентов	16
Рис. 27 – Проверка корректности адресов	17
Рис. 28 – Экспорт выводов	17
Рис. 29 – Символ системы	18
Рис. 30 – Анализ проблемных подключений.....	18
Рис. 31 – Show System with QSYS Interconnect.....	18
Рис. 32 – Schematic	19
Рис. 33 – Предустановки окна Genreration.....	19
Рис. 34 – Проверка успешности генерации HDL	19
Рис. 35 – Подключение файлов к проекту	20
Рис. 36 – Синтаксис файла lab_PD4_top.sv.....	20
Рис. 37 – Схема проекта в RTL Viewer.....	20
Рис. 38 – Тестовый файл tb_lab_PD4_top.sv	21
Рис. 39 – Тестовый файл tb_lab_PD2_top.sv	21
Рис. 40 – Моделирование проекта средствами ModelSim	22
Рис. 41 – Изменённый тестовый файл	23
Рис. 42 – Моделирование проекта средствами ModelSim (все значения).....	23

Рис. 43 – Файл для отладки модуля верхнего уровня.....	24
Рис. 44 – Схема проекта с добавлением SP_unit в RTL Viewer	24
Рис. 45 – Сигналы логического анализатора	25
Рис. 46 – Настройка окна Signal Tap II	25
Рис. 47 – Временные характеристики устройства.....	25
Рис. 48 – Настройка окна In-System Sources and Probe Editor	26
Рис. 49 – Результат SignalTap II	26

1. Задание

Средствами Platform Designer создать структуру проекта, представленную на рисунке ниже:

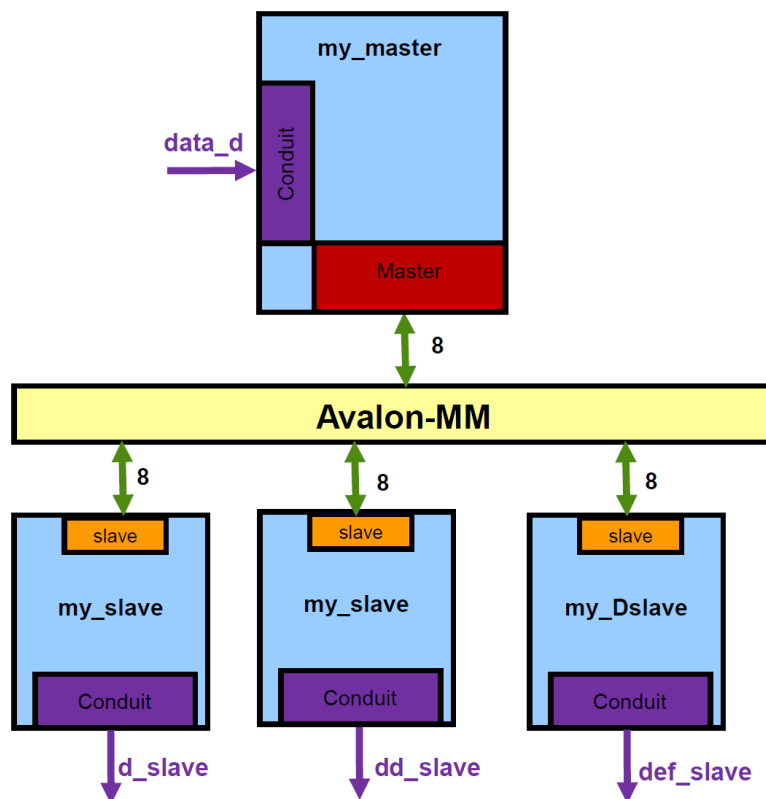


Рис. 1 – Структура проекта

Устройство, которое содержит master и 3 slave: 2 модуля **my_slave** и 1 модуль **my_Dslave** (default slave)

Master получает некоторые данные через Conduit, через 8-разрядный интерфейс мастер осуществляет адресный доступ к одному из slave'ов, настраивает соответственно slave's, либо что-то в них записывает, каждый из slave'ов имеет в себе Conduit, который помогает посмотреть на выводе slave'a то, что мы туда записали из мастера.

2. Ход работы

2.1. Создание описания модулей

Согласно структуре системы, представленной на Рис. 1, создадим описания модулей my_master, my_slave и my_Dsalve:

Создание модуля my_master

```
lab_PD4 - my_master.sv

1  `timescale 1ns/1ns
2
3  module my_master (
4      // clock and reset
5      input bit      csi_clk,          // clock clk
6      input bit      rsi_reset,        // reset reset
7      // Avalon MM master
8      output bit [7:0] avm_m0_address, // MM master address
9      output bit      avm_m0_write,    // MM master write
10     output bit [7:0] avm_m0_writedata, // MM master writedata
11     output bit      avm_m0_waitrequest, // MM master waitrequest
12     // conduit
13     input bit [7:0]  coe_c0_DA
14 );
15
16     typedef enum bit [1:0] {initSM, del1, wr1D, del2} fsm_type;
17     fsm_type fsm_MM;
18     bit [7:0] cnt_intA;
19
20     always_ff @(posedge csi_clk) begin
21         if (rsi_reset) begin
22             fsm_MM  <= initSM;
23             cnt_intA <= 8'd0;
24         end
25         else begin
26             case (fsm_MM)
27                 initSM: fsm_MM <= del1;
28                 del1:   fsm_MM <= wr1D;
29                 wr1D:   if (avm_m0_waitrequest)
30                         fsm_MM <= wr1D;
31                         else
32                             fsm_MM <= del2;
33                 del2:   begin
34                         fsm_MM <= initSM;
35                         cnt_intA <= cnt_intA + 8'd1;
36                     end
37             endcase
38         end
39     end
40
41     always_comb begin
42         case (fsm_MM)
43             wr1D:
44                 begin
45                     avm_m0_address  = cnt_intA;
46                     avm_m0_write    = 1'd1;
47                     avm_m0_writedata = cnt_intA + coe_c0_DA;
48                 end
49             default:
50                 begin
51                     avm_m0_address  = 8'd255;
52                     avm_m0_write    = 1'd0;
53                     avm_m0_writedata = 8'd255;
54                 end
55             endcase
56         end
57     endmodule
58
```

Рис. 2 – Модуль my_master

Модуль `my_master` функционирует как главное устройство в системе Avalon Memory-Mapped (MM). Он управляет передачей данных от мастера к другим компонентам.

Модуль работает на основе конечного автомата (FSM), который. FSM имеет четыре состояния: `initSM`, `del1`, `wr1D`, `del2`.

- `initSM`: Начальное состояние.
- `del1`: Задержка для ожидания данных от мастера Avalon MM (чтобы отделить циклы записи по шине, это не обязательно, но так будет наглядно при просмотре waveform).
- `wr1D`: Ожидание завершения операции записи данных от мастера Avalon MM.
- `del2`: Дополнительная задержка после завершения операции записи.

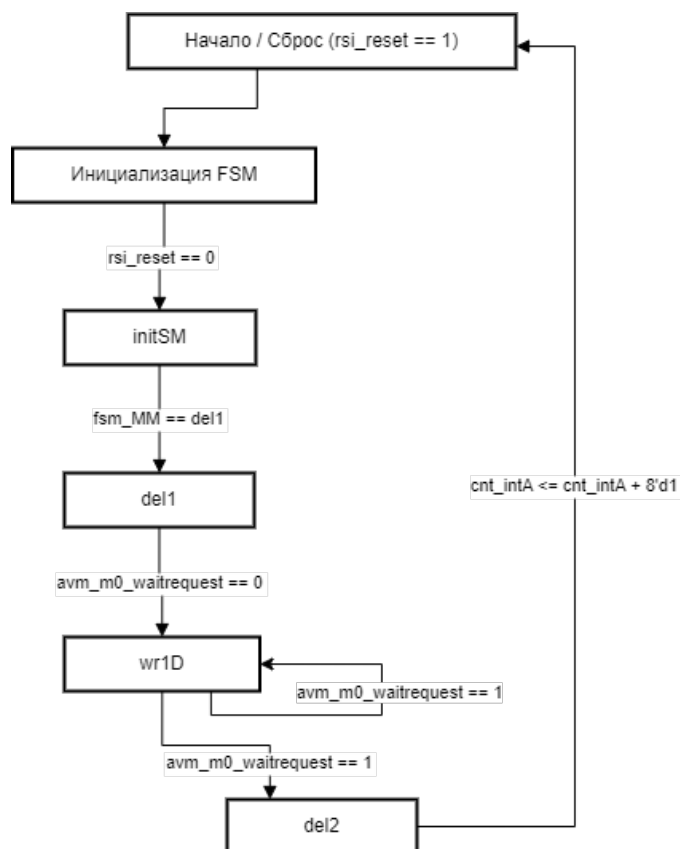


Рис. 3 – Схема конечного автомата

В состоянии `wr1D`, когда мастер отправляет данные на запись, модуль устанавливает значения выходных сигналов `avm_m0_address`, `avm_m0_write`, `avm_m0_writedata` в соответствии с требуемыми операциями записи.

Пока операция записи выполняется (`avm_m0_waitrequest == 1`), FSM остаётся в состоянии `wr1D`, ожидая окончания операции записи.

Когда операция записи завершается (когда `avm_m0_waitrequest == 0`), FSM переходит в состояние `del2`, где инкрементируется счётчик `cnt_intA`.

Значение счётчика `cnt_intA` выводится через сигнал `coe_c0_DA` для передачи его другим компонентам через интерфейс Conduit.

*Conduit обеспечивает канал связи между различными компонентами системы, что позволяет им обмениваться данными и взаимодействовать друг с другом.

Таким образом, модуль `my_master` обеспечивает правильную передачу данных от мастера к другим устройствам в системе (`my_slave` и `my_Dslave`).

Схема master средствами RTL Viewer будет выглядеть следующим образом:

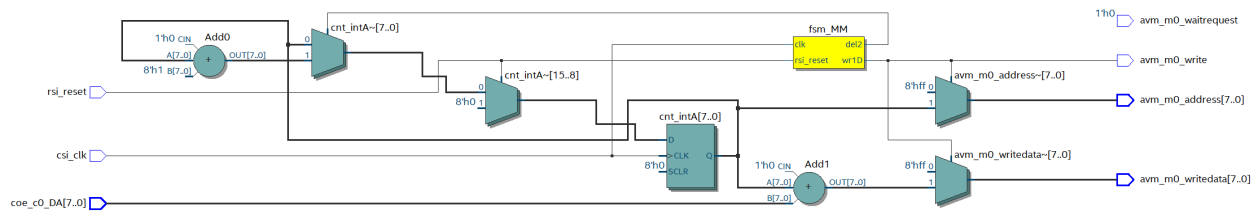


Рис. 4 – Модуль my_master в RTL Viewer

Также, откроем State Machine Viewer и убедимся в правильности построенного автомата:

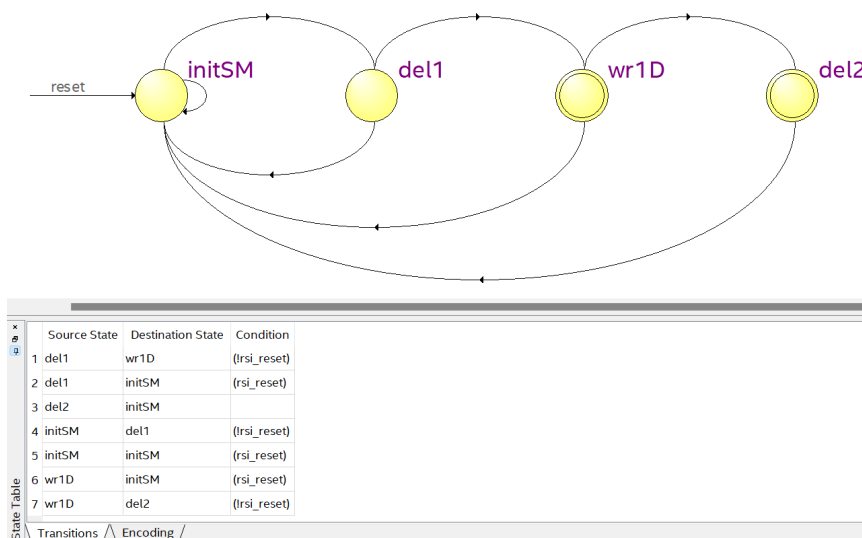


Рис. 5 – Схема конечного автомата в State Machine Viewer

Схема конечного автомата удовлетворяет той, что представлена на Рис. 3.

Создание модуля my_slave

```
lab_PD4 - my_slave.sv

1 `timescale 1ns/1ns
2 module my_slave (
3   // clock and reset
4   input bit      csi_clk, // clock clk
5   input bit      rsi_reset, // reset reset
6   // MM Slave
7   input bit [7:0] avs_s0_writedata, // MM Slave writedata
8   input bit      avs_s0_write, // MM Slave write
9   output bit      avs_s0_waitrequest, // MM Slave waitrequest
10  // Conduit
11  output bit [7:0] coe_s0_Dout
12 );
13 bit [7:0] rg_DATA;
14 assign avs_s0_waitrequest = 1'b0;
15
16 always_ff @(posedge csi_clk) begin
17   if (rsi_reset)
18     rg_DATA <= 8'd0;
19   else if (avs_s0_write)
20     rg_DATA <= avs_s0_writedata;
21 end
22 assign coe_s0_Dout = rg_DATA;
23 endmodule
24
```

Рис. 6 – Модуль my_slave

Модуль my_slave в интерфейсе Avalon Memory-Mapped (MM) функционирует как подчинённое устройство, принимая данные от мастера и передавая их через выходной сигнал coe_s0_Dout.

Он использует тактовый сигнал csi_clk для синхронизации операций и сигнал сброса rsi_reset для инициализации внутренних состояний.

Модуль содержит 8-битный регистр данных rg_DATA, который обновляется при каждом положительном фронте csi_clk, если активирован сигнал записи avs_s0_write. При активации сигнала сброса регистр rg_DATA сбрасывается в ноль.

Данные, хранящиеся в регистре rg_DATA, передаются через выходной сигнал coe_s0_Dout. Сигнал avs_s0_waitrequest всегда устанавливается в ноль, что означает отсутствие запроса на ожидание со стороны подчинённого устройства.

Посмотрим, как выглядит диаграмма этого модуля в RTL Viewer:

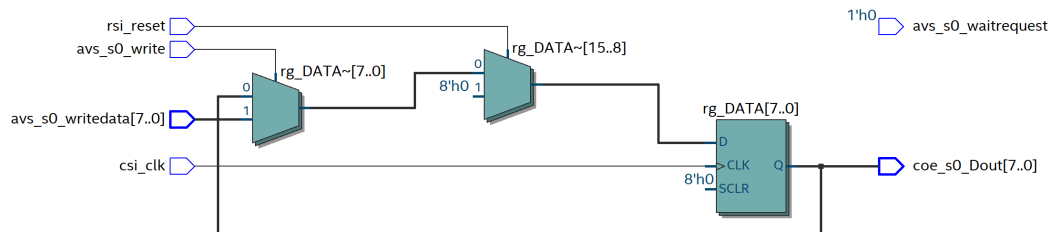


Рис. 7 – Модуль my_slave в RTL Viewer

Создание модуля my_Dslave

```
lab_PD4 - my_Dslave.vv
1 `timescale 1ns/1ns
2 module my_Dslave (
3 // clock and reset
4     input bit        csi_clk, // clock clk
5     input bit        rsi_reset, // reset reset
6 // MM Slave
7     input bit [7:0]  avs_s0_writedata, // MM Slave writedata
8     input bit        avs_s0_write, // MM Slave write
9     output bit       avs_s0_waitrequest, // MM Slave waitrequest
10 // Conduit
11     output bit [7:0] coe_s0_Dout
12 );
13     bit [7:0] cnt_;
14     assign avs_s0_waitrequest = 1'b0;
15
16     always_ff @(posedge csi_clk) begin
17         if (rsi_reset)
18             cnt_ <= 8'd0;
19         else if (avs_s0_write)
20             cnt_ <= cnt_ + 8'd1;
21     end
22     assign coe_s0_Dout = cnt_;
23 endmodule
24
```

Рис. 8 – Модуль my_Dslave

Модуль my_Dslave является простым устройством в системе, которое принимает данные от мастера Avalon MM и передаёт их через интерфейс Conduit. Когда мастер отправляет данные, my_Dslave сохраняет их во внутреннем регистре и затем передаёт через выходной порт coe_s0_Dout через интерфейс Conduit без каких-либо изменений. Это позволяет эффективно передавать данные от мастера Avalon MM к другим частям системы, используя my_Dslave в качестве посредника, без необходимости дополнительной обработки или изменений данных.

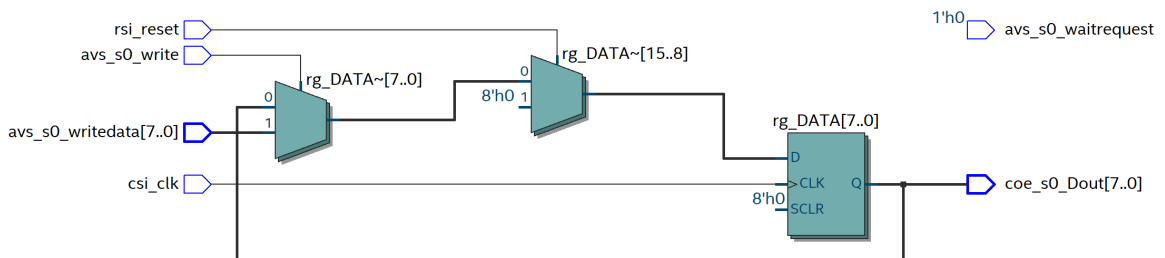


Рис. 9 – Модуль my_slave в RTL Viewer

Отличие my_slave от my_Dslave

Модули my_slave и my_Dslave оба выполняют функцию обработки данных в системе, но есть несколько ключевых различий между ними.

Если обычный slave просто записывает то, что пришло на шину write_data, то Dslave будет хранить число обращений к нему

Принцип работы системы

Таким образом, автомат берёт данные с Conduit'a, выдает их на шину master (с 0 по 255 такты цикла), м/у записями (когда не write), формирует признаки того, что он не работает, выдавая 255, когда работает, выдаёт 1.

2.2. Создание проекта

Начало работы в PD

Откроем PD и сохраним систему:

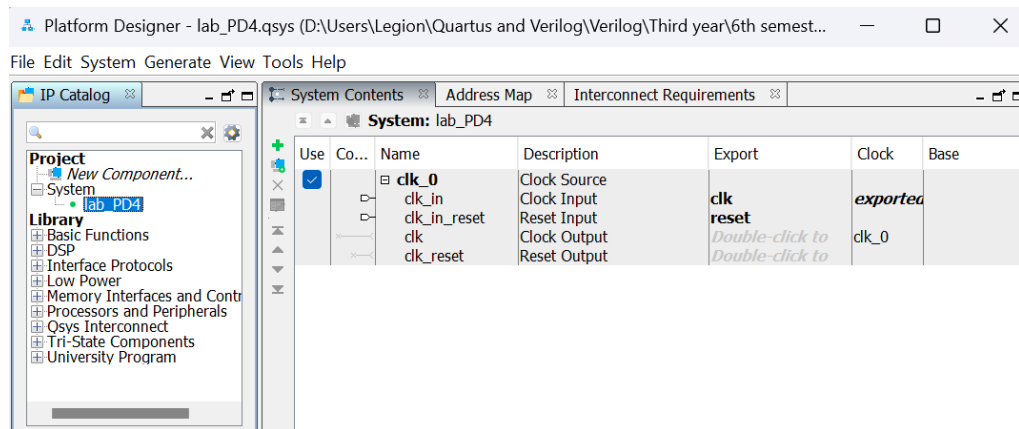


Рис. 10 – Исходное окно PD

Через строчку New Component импортируем модули, созданные ранее (my_master, my_slave, my_Dslave):

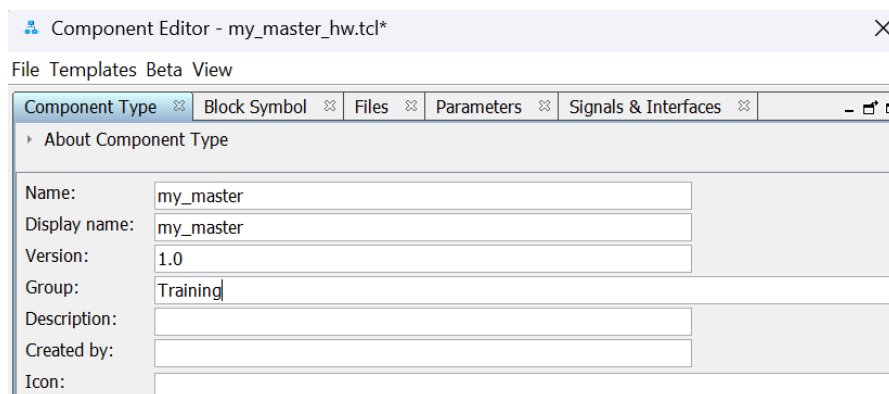


Рис. 11 – Закладка Component Type в New Component

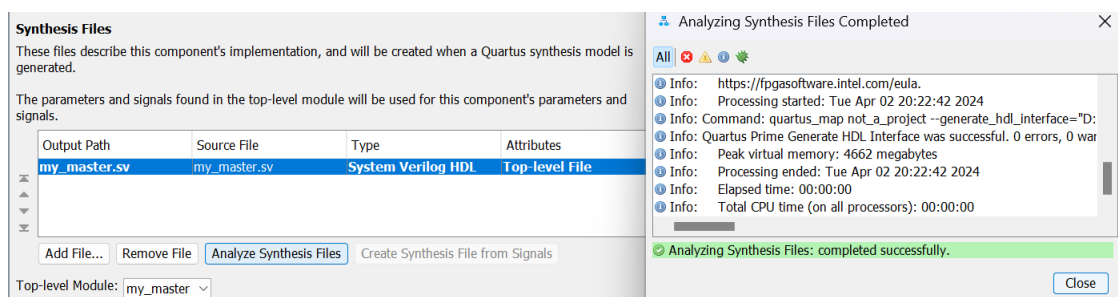


Рис. 12 – Закладка Files в New Component

Verilog Simulation Files			
These files will be produced when a Verilog simulation model is generated.			
Output Path	Source File	Type	Attributes
my_master.vv	my_master.vv	System Verilog HDL	no attributes
<div>Add File...</div> <div>Remove File</div> <div>Copy from Synthesis Files</div>			

Рис. 13 – Выбор файла для симуляции

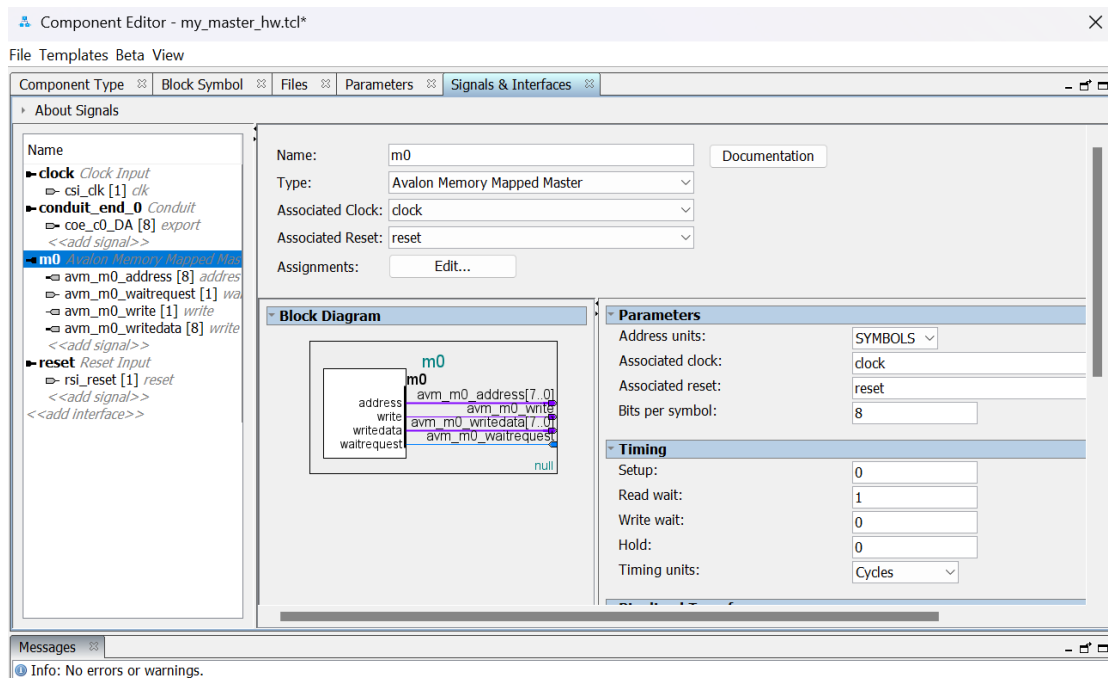


Рис. 14 – Закладка Signals & Interfaces в New Component (master)

Убедимся в том, что модуль my_master был успешно добавлен в проект:

System: lab_PD4						
Use	Co...	Name	Description	Export	Clock	Base
<input checked="" type="checkbox"/>		clk_0	Clock Source			
		clk_in	Clock Input	clk	exported	
		clk_in_reset	Reset Input	reset		
		clk	Clock Output		clk_0	
		clk_reset	Reset Output			

Рис. 15 – Проверка успешного добавления мастера в проект

Теперь проведём аналогичные действия для модулей my_slave и my_Dslave:

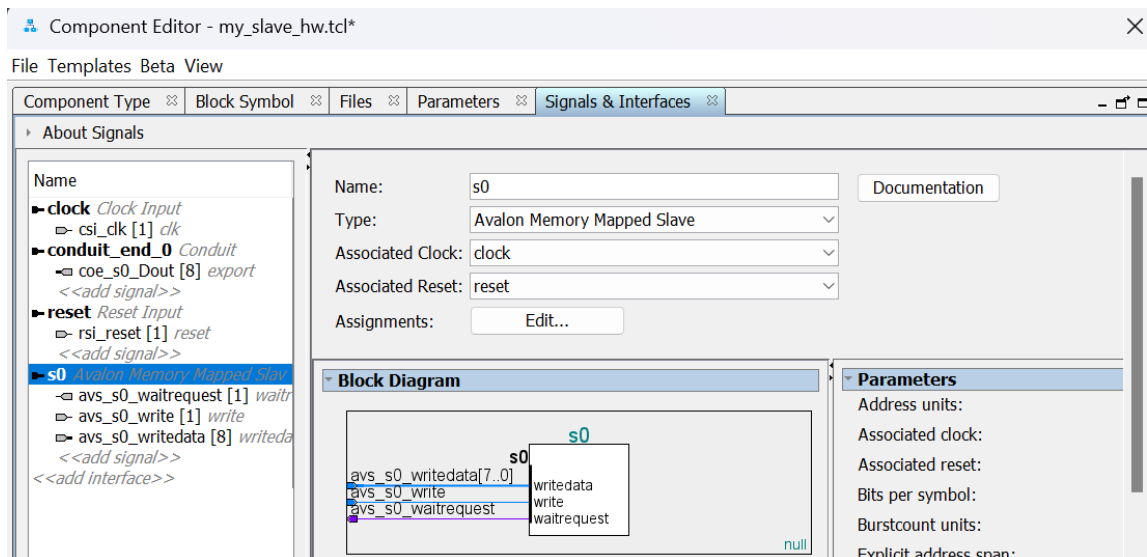


Рис. 16 – Закладка Signals & Interfaces в New Component (slave)

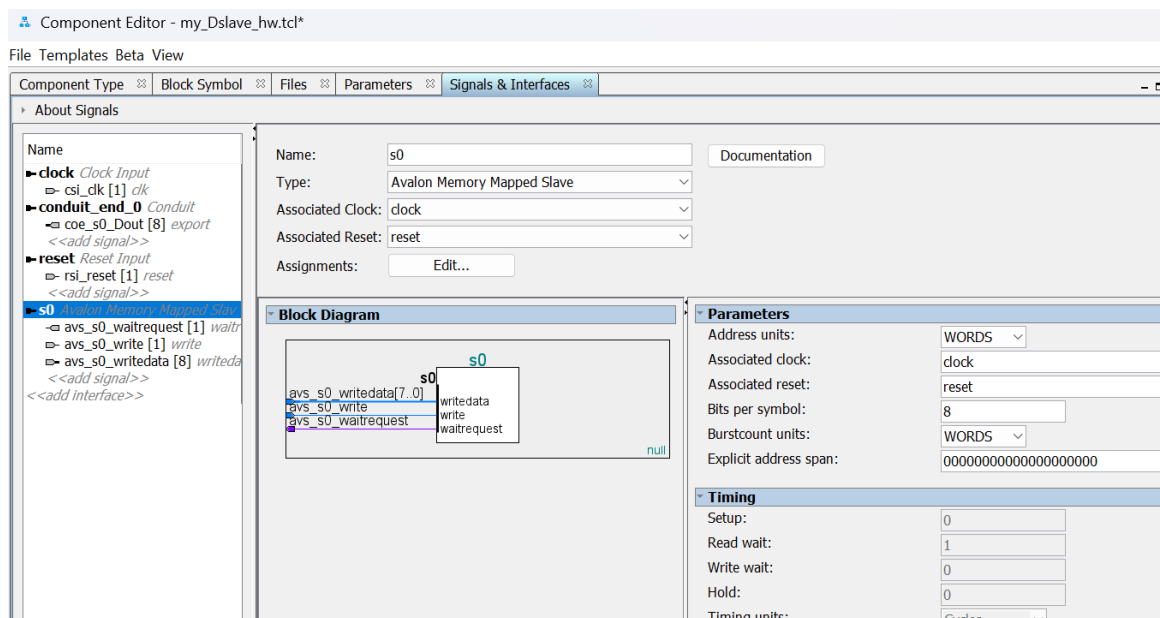


Рис. 17 – Закладка Signals & Interfaces в New Component (Dslave)

Убедимся, что все модули успешно добавились в систему:

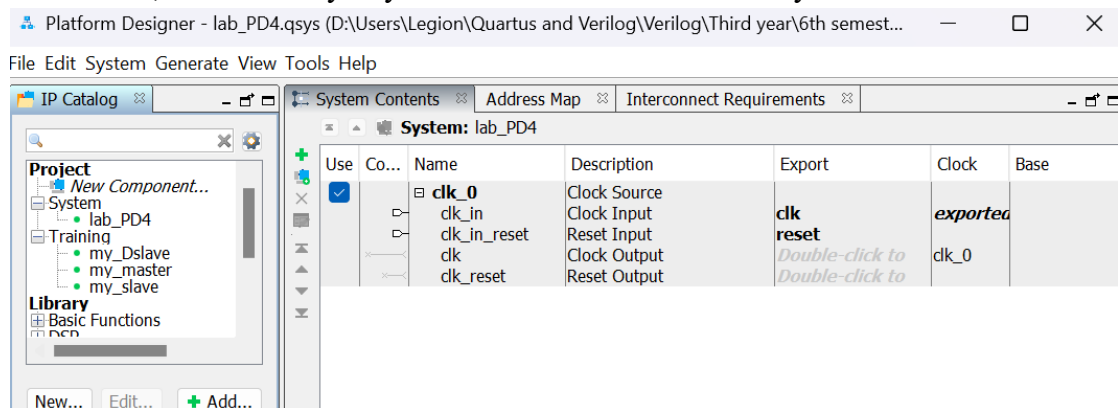


Рис. 18 – Проверка успешности добавления модулей

Согласно схеме устройства на Рис. 1 добавим необходимые модули из тех, которые были только что нами созданы и проверим, что система выглядит следующим образом:

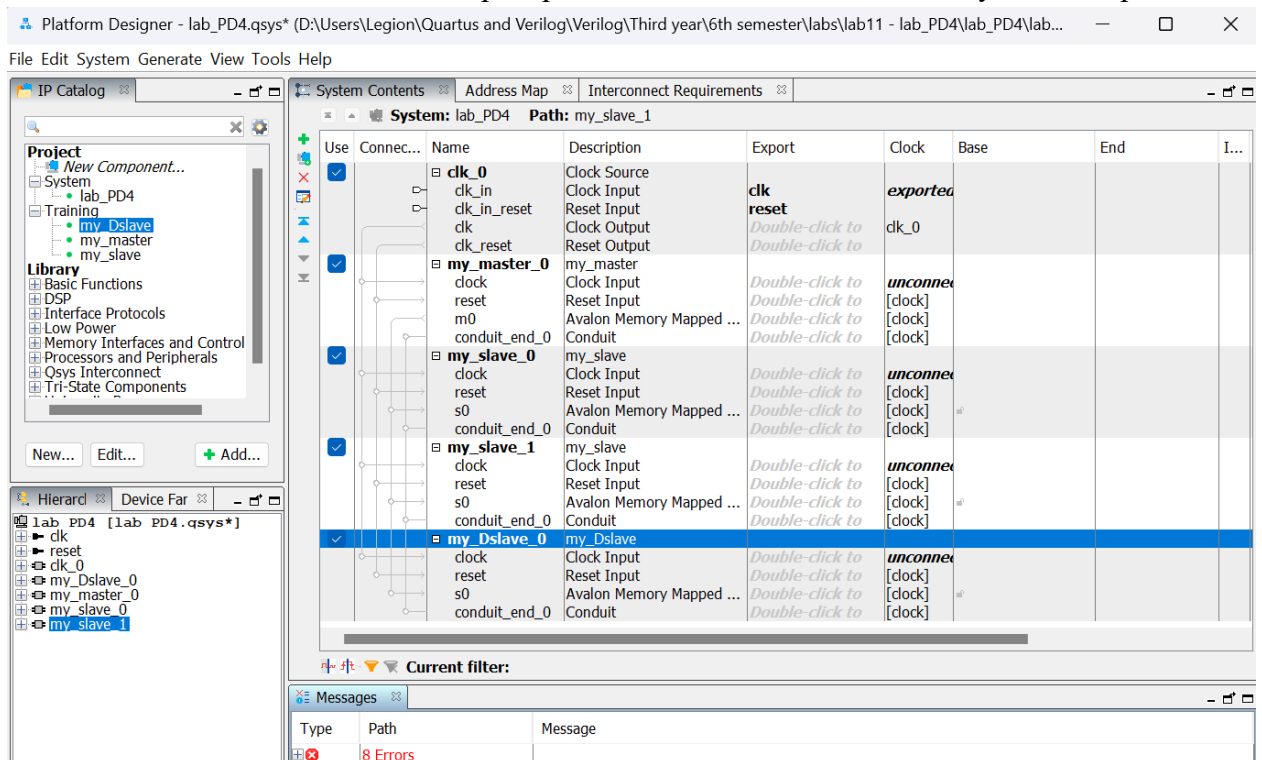


Рис. 19 – Добавление модулей в систему

2.3. Настройка сигналов

Настройка clk

Зададим значение Reset synchronous edges = Deassert

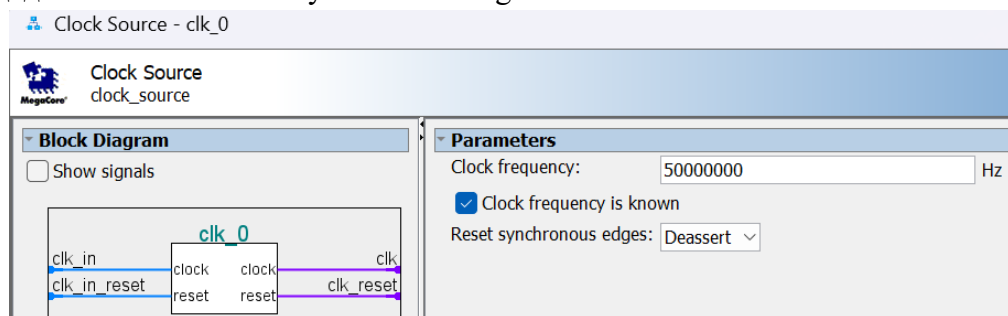


Рис. 20 – Настройка компонента clk

2.4. Подключение сигналов

Подключение тактового сигнала

Выделим интерфейс clk компонента clk_0, и, открыв его соединения, выберем подключение ко всем тактовым входам:

System Contents	Address Map	Interconnect Requirements	Connections	
System: lab_PD4 Path: clk_0.clk				
Connected to: clk_0.clk				
Connected	Connection	Clock Crossing	Data Width	Burst
	clk_0.clk/my_Dslave_0.clock			
	clk_0.clk/my_master_0.clock			
	clk_0.clk/my_slave_0.clock			
	clk_0.clk/my_slave_1.clock			

Рис. 21 – Подключение тактового сигнала (1)

Переименуем сигналы и выполним Filter → Clock and Reset Interfaces, убедимся, что соединения выполнены корректно:

System Contents		Address Map		Interconnect Requirements		Connections			
System: lab_PD4 Path: clk.clk									
Use	Co...	Name	Description	Export	Clock	Base	End	I...	Ta
		clk	Clock Source						
		clk_in	Clock Input	clk	exported				
		clk_in_reset	Reset Input	reset	[clk_in]				
		clk	Clock Output	Double-click to	clk				
		clk_reset	Reset Output	Double-click to	clk				
		my_master	my_master						
		clock	Clock Input	Double-click to	clk				
		reset	Reset Input	Double-click to	[clock]				
		my_slave_1	my_slave						
		clock	Clock Input	Double-click to	clk				
		reset	Reset Input	Double-click to	[clock]				
		my_slave_2	my_slave						
		clock	Clock Input	Double-click to	clk				
		reset	Reset Input	Double-click to	[clock]				
		my_Dslave	my_Dslave						
		clock	Clock Input	Double-click to	clk				
		reset	Reset Input	Double-click to	[clock]				

Рис. 22 – Подключение тактового сигнала (2)

Подключение сигнала Reset

Выполним System → Create Global Reset Network и убедимся, что система выглядит корректно, сигнал Reset подключен:

System Contents		Address Map		Interconnect Requirements		Connections			
System: lab_PD4 Path: clk.clk									
Use	Co...	Name	Description	Export	Clock	Base	End	I...	Ta
<input checked="" type="checkbox"/>		<div>clk</div> <div>clk_in</div> <div>clk_in_reset</div>	Clock Source Clock Input Reset Input	clk reset	exported [clk_in]				
<input checked="" type="checkbox"/>		<div>clk</div> <div>clk_reset</div>	Clock Output Reset Output	Double-click to Double-click to	clk clk				
<input checked="" type="checkbox"/>		<div>my_master</div> <div>clock</div> <div>reset</div>	my_master Clock Input Reset Input	Double-click to Double-click to	clk [clock]				
<input checked="" type="checkbox"/>		<div>my_slave_1</div> <div>clock</div> <div>reset</div>	my_slave Clock Input Reset Input	Double-click to Double-click to	clk [clock]	ai			
<input checked="" type="checkbox"/>		<div>my_slave_2</div> <div>clock</div> <div>reset</div>	my_slave Clock Input Reset Input	Double-click to Double-click to	clk [clock]	ai			
<input checked="" type="checkbox"/>		<div>my_Dslave</div> <div>clock</div> <div>reset</div>	my_Dslave Clock Input Reset Input	Double-click to Double-click to	clk [clock]	ai			

Рис. 23 – Подключение сигнала Reset

2.5. Подключение Avalon-MM интерфейсов

Выполним Filter → Avalon-MM Interfaces и выберем соединения так, как показано на картинке ниже

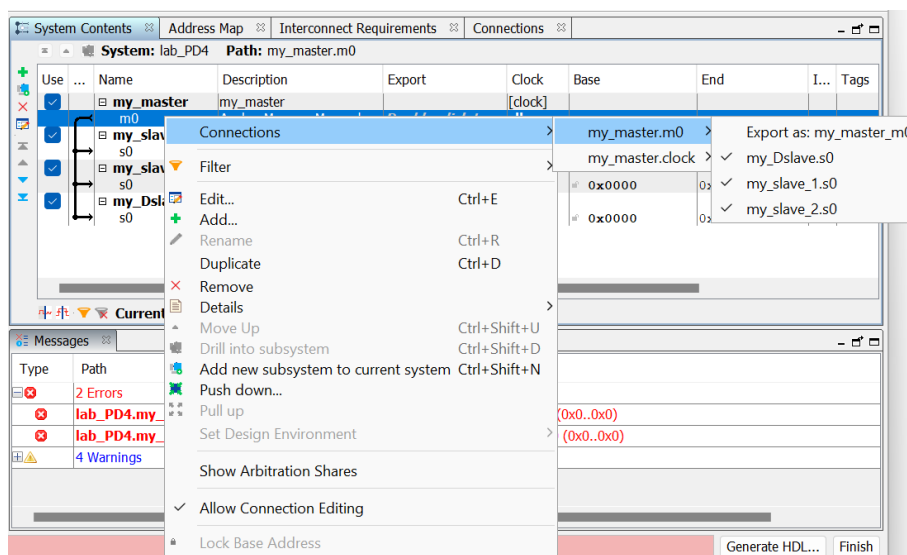


Рис. 24 – Подключение Avalon-MM интерфейсов

В столбце Default Slave (добавлен через ПКМ), поставив галочку, тем самым выполнив check box:

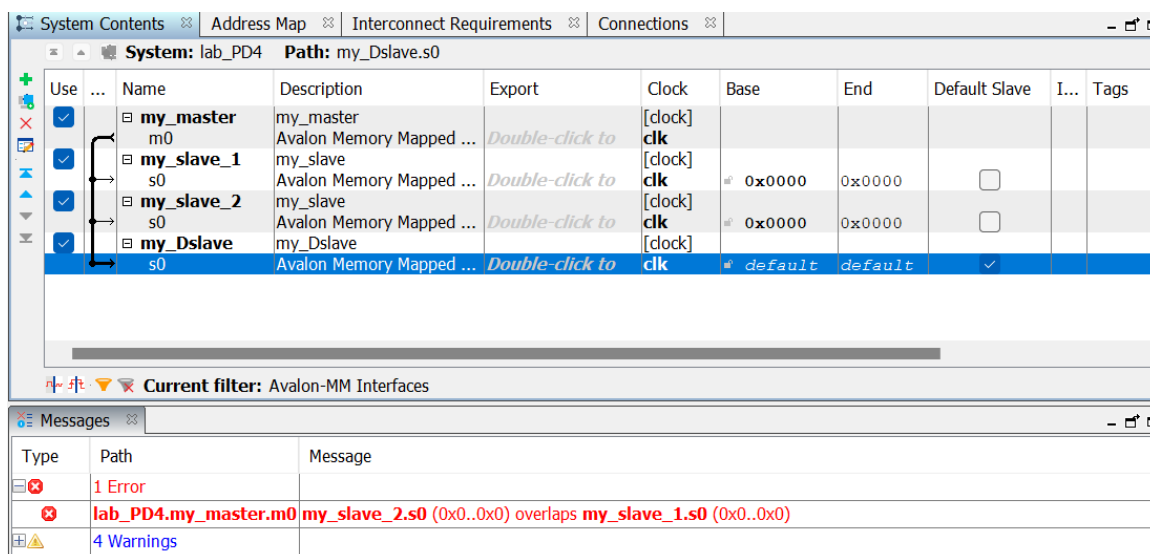


Рис. 25 – Фиксация адресов

Компоненту my_slave_1.s0 назначим базовый адрес = 1, а компоненту my_slave_2.s0 – базовый адрес = 2. После этого зафиксируем адреса (замочек), система будет выглядеть следующим образом:

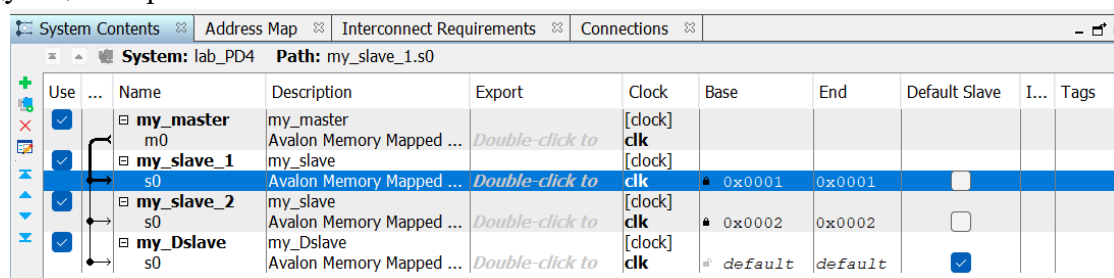


Рис. 26 – Назначение правильных адресов для компонентов

Вкладка Address Map будет выглядеть следующим образом:

System Contents	Address Map	Interconnect Requirements
System: lab_PD4 Path: my_Dslave		
	my_master.m0	
my_Dslave.s0	default	
my_slave_1.s0	0x0001 - 0x0001	
my_slave_2.s0	0x0002 - 0x0002	

Рис. 27 – Проверка корректности адресов

Запомним базовые адреса:

- my_slave_1.s0 = 1,
- my_slave_2.s0 = 2.

*Позже он будут указываться при настройке модулей my_slave_1 и my_slave_2 соответственно

2.6. Экспорт выводов

Проведём экспорт выводов путём задания имён для выделенных модулей в столбце Export (значения data_d, d_slave, did_slave и def_slave в соответствующем столбце выделенных строк):

System Contents		Address Map		Interconnect Requirements		Connections				
System: lab_PD4 Path: my_master.conduit_end_0										
Use	Connec...	Name	Description	Export	Clock	Base	End	Default Slave	I...	Tags
<input checked="" type="checkbox"/>		<div>clk</div>	Clock Source							
<input checked="" type="checkbox"/>		<div>clk_in</div>	Clock Input	clk	clk	exported				
<input checked="" type="checkbox"/>		<div>clk_in_reset</div>	Reset Input	clk	reset	[clk_in]				
<input checked="" type="checkbox"/>		<div>clk</div>	Clock Output	clk	Double-click to	clk				
<input checked="" type="checkbox"/>		<div>clk_reset</div>	Reset Output	clk	Double-click to	clk				
<input checked="" type="checkbox"/>		<div>my_master</div>	my_master	my_master	Double-click to	clk				
<input checked="" type="checkbox"/>		<div>dclk</div>	Clock Input	my_master	Double-click to	[dclk]				
<input checked="" type="checkbox"/>		<div>reset</div>	Reset Input	my_master	Double-click to	[dclk]				
<input checked="" type="checkbox"/>		<div>m0</div>	Avalon Memory Mapped ...	my_master	Double-click to	[dclk]				
<input checked="" type="checkbox"/>		<div>conduit_end_0</div>	Conduit	my_slave	data_d	[dclk]				
<input checked="" type="checkbox"/>	<div>my_slave_1</div>	my_slave	my_slave	Double-click to	clk					
<input checked="" type="checkbox"/>	<div>dclk</div>	Clock Input	my_slave	Double-click to	[dclk]					
<input checked="" type="checkbox"/>	<div>reset</div>	Reset Input	my_slave	Double-click to	[dclk]					
<input checked="" type="checkbox"/>	<div>s0</div>	Avalon Memory Mapped ...	my_slave	Double-click to	[dclk]	0x0001	0x0001	<input type="checkbox"/>		
<input checked="" type="checkbox"/>	<div>conduit_end_0</div>	Conduit	my_slave	d_slave	[dclk]					
<input checked="" type="checkbox"/>	<div>my_slave_2</div>	my_slave	my_slave	Double-click to	clk					
<input checked="" type="checkbox"/>	<div>dclk</div>	Clock Input	my_slave	Double-click to	[dclk]					
<input checked="" type="checkbox"/>	<div>reset</div>	Reset Input	my_slave	Double-click to	[dclk]					
<input checked="" type="checkbox"/>	<div>s0</div>	Avalon Memory Mapped ...	my_slave	Double-click to	[dclk]	0x0002	0x0002	<input type="checkbox"/>		
<input checked="" type="checkbox"/>	<div>conduit_end_0</div>	Conduit	my_Dslave	dd_slave	[dclk]					
<input checked="" type="checkbox"/>	<div>my_Dslave</div>	my_Dslave	my_Dslave	Double-click to	clk					
<input checked="" type="checkbox"/>	<div>dclk</div>	Clock Input	my_Dslave	Double-click to	[dclk]					
<input checked="" type="checkbox"/>	<div>reset</div>	Reset Input	my_Dslave	Double-click to	[dclk]					
<input checked="" type="checkbox"/>	<div>s0</div>	Avalon Memory Mapped ...	my_Dslave	Double-click to	[dclk]	* default	default	<input checked="" type="checkbox"/>		
<input checked="" type="checkbox"/>	<div>conduit_end_0</div>	Conduit	my_Dslave	def_slave	[dclk]					

Рис. 28 – Экспорт выводов

Проверим, что окно сообщений не содержит ошибок и пре

2.7. Анализ системы

Проверка блока

Выполним View → Block Symbol и убедимся в том, что символ системы построен правильно:

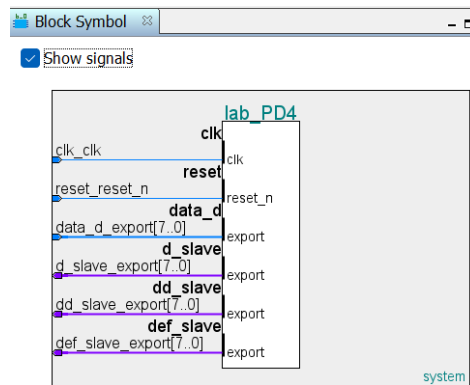


Рис. 29 – Символ системы

Выполним View → Clock domains Beta, выберем режим отображения Reset. Заметим, что проблемных подключений не выявлено:

Use	Connec...	Name	Description	Export	Clock	Base	End	Default Slave	I...	Tags
<input checked="" type="checkbox"/>		clk	Clock Source	clk	exported					
<input checked="" type="checkbox"/>		clk_in	Clock Input	reset	[clk_in]					
<input checked="" type="checkbox"/>		clk_in_reset	Reset Input	reset	[clk_in]					
<input checked="" type="checkbox"/>		clk	Clock Output	clk	clk					
<input checked="" type="checkbox"/>		clk_reset	Reset Output	clk	clk					
<input checked="" type="checkbox"/>		my_master	my_master							
<input checked="" type="checkbox"/>		clock	Clock Input	clk	[clock]					
<input checked="" type="checkbox"/>		reset	Reset Input	reset	[clock]					
<input checked="" type="checkbox"/>		m0	Avalon Memory Mapped Master	data_d	[clock]					
<input checked="" type="checkbox"/>		conduit_end_0	Conduit	data_d	[clock]					
<input checked="" type="checkbox"/>		my_slave_1	my_slave							
<input checked="" type="checkbox"/>		clock	Clock Input	clk	[clock]					
<input checked="" type="checkbox"/>		reset	Reset Input	reset	[clock]					
<input checked="" type="checkbox"/>		s0	Avalon Memory Mapped Slave	d_slave	[clock]	# 0x0001	0x0001	<input type="checkbox"/>		
<input checked="" type="checkbox"/>		conduit_end_0	Conduit	d_slave	[clock]					
<input checked="" type="checkbox"/>		my_slave_2	my_slave							
<input checked="" type="checkbox"/>		clock	Clock Input	clk	[clock]					
<input checked="" type="checkbox"/>		reset	Reset Input	reset	[clock]					
<input checked="" type="checkbox"/>		s0	Avalon Memory Mapped Slave	dd_slave	[clock]	# 0x0002	0x0002	<input type="checkbox"/>		
<input checked="" type="checkbox"/>		conduit_end_0	Conduit	dd_slave	[clock]					
<input checked="" type="checkbox"/>		my_Dslave	my_Dslave							
<input checked="" type="checkbox"/>		clock	Clock Input	clk	[clock]					
<input checked="" type="checkbox"/>		reset	Reset Input	reset	[clock]					
<input checked="" type="checkbox"/>		s0	Avalon Memory Mapped Slave	def_slave	[clock]	# default	default	<input checked="" type="checkbox"/>		
<input checked="" type="checkbox"/>		conduit_end_0	Conduit	def_slave	[clock]					

Рис. 30 – Анализ проблемных подключений

Выполним команду System → Show System with PD Interconnect (Show System with QSYS Interconnect). Проверим, был добавлен только модуль mm_interconnect_0.

Use	Connections	Name	Description	Export	Clock	Base	End	I...	Default Slave	Tags
<input checked="" type="checkbox"/>		mm_intercon...	MM Interconnect							
<input checked="" type="checkbox"/>		clk_clk	Clock Input	clk	[clk_clk]	# 0x0000	0x00ff	<input type="checkbox"/>		
<input checked="" type="checkbox"/>		my_master_re...	Reset Input	reset	[clk_clk]					
<input checked="" type="checkbox"/>		my_master_m0	Avalon Memory Mapped Master	data_d	[clk_clk]					
<input checked="" type="checkbox"/>		my_Dslave_s0	Avalon Memory Mapped Master	d_slave	[clk_clk]					
<input checked="" type="checkbox"/>		my_slave_1_s0	Avalon Memory Mapped Master	dd_slave	[clk_clk]					
<input checked="" type="checkbox"/>		my_slave_2_s0	Avalon Memory Mapped Master	def_slave	[clk_clk]					
<input checked="" type="checkbox"/>		clk	Clock Source	clk	exported					
<input checked="" type="checkbox"/>		clk_in	Clock Input	reset	[clk_in]					
<input checked="" type="checkbox"/>		clk_in_reset	Reset Input	reset	[clk_in]					
<input checked="" type="checkbox"/>		clk	Clock Output	clk	clk					
<input checked="" type="checkbox"/>		clk_reset	Reset Output	clk	clk					
<input checked="" type="checkbox"/>		my_master	my_master							
<input checked="" type="checkbox"/>		clock	Clock Input	clk	[clock]					
<input checked="" type="checkbox"/>		reset	Reset Input	reset	[clock]					
<input checked="" type="checkbox"/>		m0	Avalon Memory Mapped Master	data_d	[clock]					
<input checked="" type="checkbox"/>		conduit_end_0	Conduit	data_d	[clock]					
<input checked="" type="checkbox"/>		my_slave_1	my_slave							
<input checked="" type="checkbox"/>		clock	Clock Input	clk	[clock]					
<input checked="" type="checkbox"/>		reset	Reset Input	reset	[clock]					
<input checked="" type="checkbox"/>		s0	Avalon Memory Mapped Slave	d_slave	[clock]	# 0x0000	0x0000	<input type="checkbox"/>		
<input checked="" type="checkbox"/>		conduit_end_0	Conduit	d_slave	[clock]					
<input checked="" type="checkbox"/>		my_slave_2	my_slave							
<input checked="" type="checkbox"/>		clock	Clock Input	clk	[clock]					
<input checked="" type="checkbox"/>		reset	Reset Input	reset	[clock]					
<input checked="" type="checkbox"/>		s0	Avalon Memory Mapped Slave	dd_slave	[clock]	# 0x0000	0x0000	<input type="checkbox"/>		
<input checked="" type="checkbox"/>		conduit_end_0	Conduit	dd_slave	[clock]					
<input checked="" type="checkbox"/>		my_Dslave	my_Dslave							
<input checked="" type="checkbox"/>		clock	Clock Input	clk	[clock]					
<input checked="" type="checkbox"/>		reset	Reset Input	reset	[clock]					
<input checked="" type="checkbox"/>		s0	Avalon Memory Mapped Slave	def_slave	[clock]	# 0x0000	0x0000	<input type="checkbox"/>		
<input checked="" type="checkbox"/>		conduit_end_0	Conduit	def_slave	[clock]					

Рис. 31 – Show System with QSYS Interconnect

Анализ с помощью Schematic

Выполним View → Schematic, в качестве фильтра введём in и убедимся в том, что система синхронизации и каналы ST системы подключены верно:

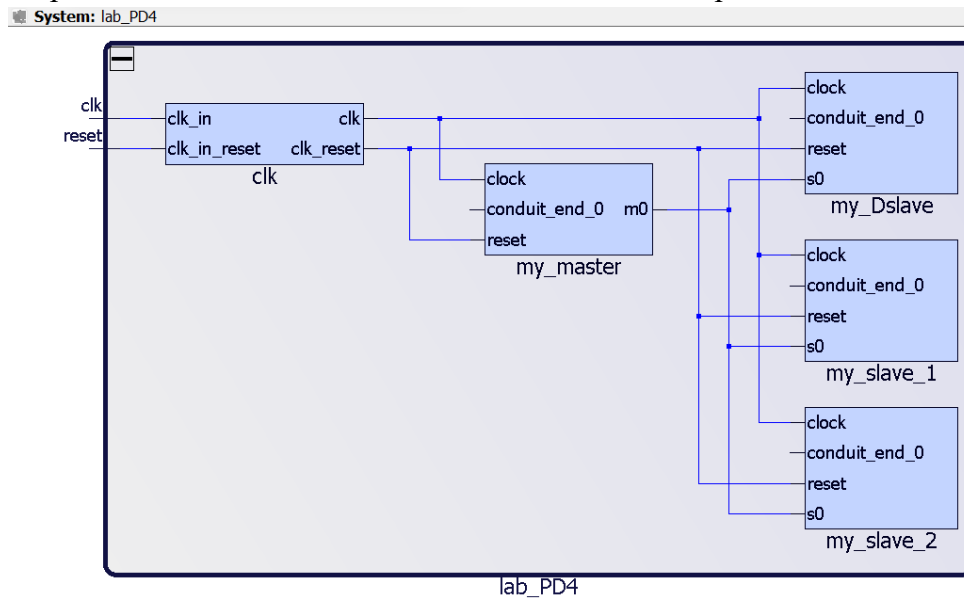


Рис. 32 – Schematic

Генерация системы

Выполним PD → Generate HDL и укажем следующие предустановки для генерации:

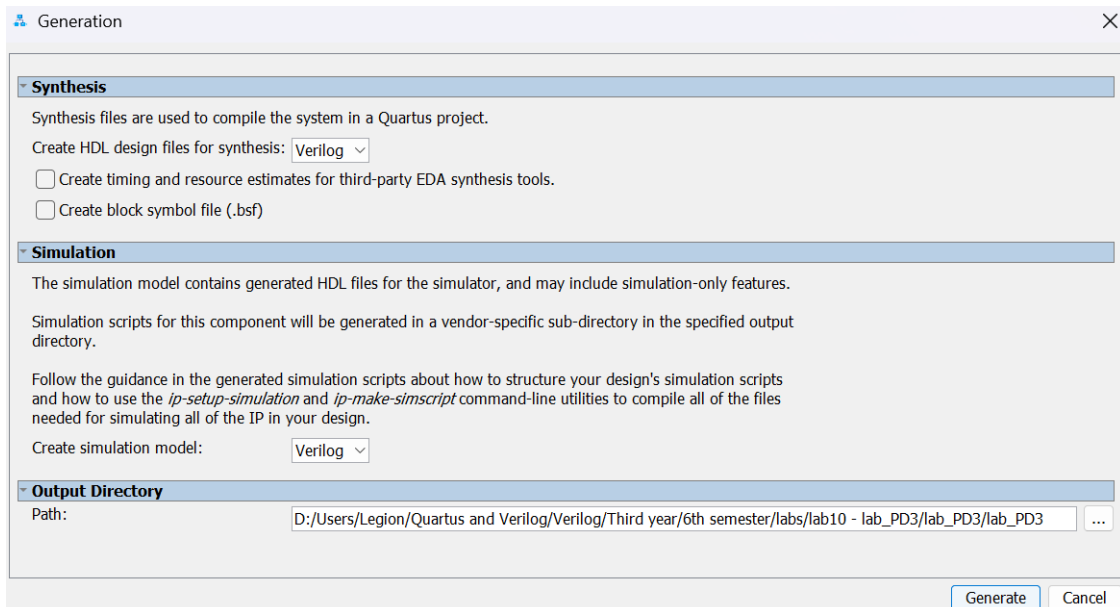


Рис. 33 – Предустановки окна Generation

Удостоверимся в том, что генерация прошла успешно:

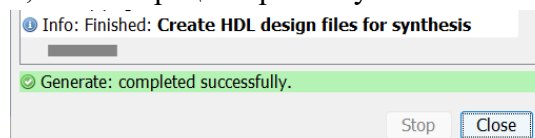


Рис. 34 – Проверка успешности генерации HDL

2.8. Подключение файлов к проекту

Подключим файлы к проекту в Quartus

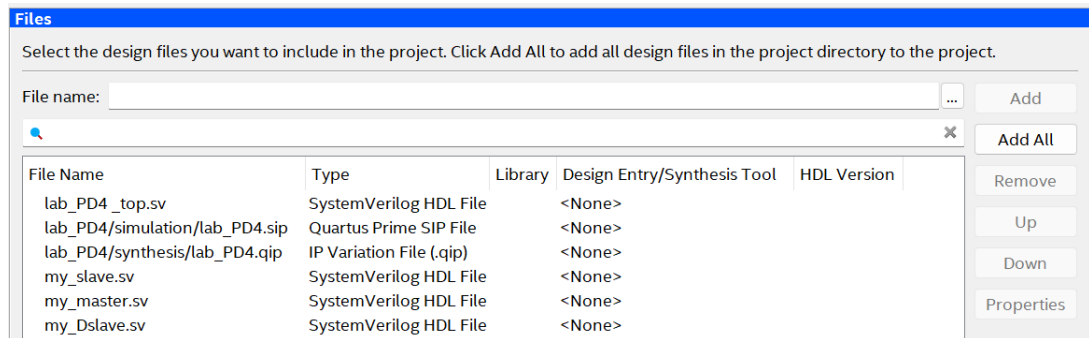


Рис. 35 – Подключение файлов к проекту

Синтаксис файла lab_PD4_top.v:

```
lab_PD4 - lab_PD4_top.v

1 `timescale 1ns/1ns
2 module lab_PD4_top (
3     input bit clk,
4     input bit reset,
5     input bit [7:0] data_d,
6     output bit [7:0] dd_slave,
7     output bit [7:0] d_slave,
8     output bit [7:0] def_slave,
9 );
10 lab_PD4 lab4_sys_inst (
11     .clk_clk (clk),
12     .reset_reset_n (reset),
13     .def_slave_export (def_slave),
14     .dd_slave_export (d_slave),
15     .d_slave_export (d_slave),
16     .data_d_export (data_d)
17 );
18 endmodule
19
```

Рис. 36 – Синтаксис файла lab_PD4_top.v

Выполним анализ и синтез проекта средствами QP и убедимся в правильности схемы средствами RTL Viewer:

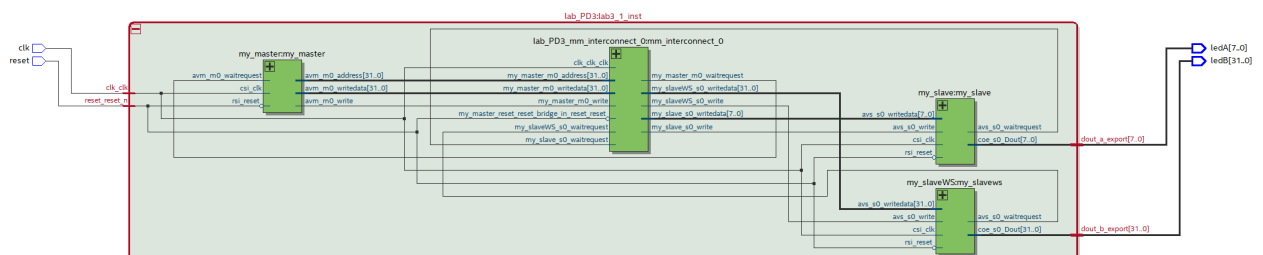


Рис. 37 – Схема проекта в RTL Viewer

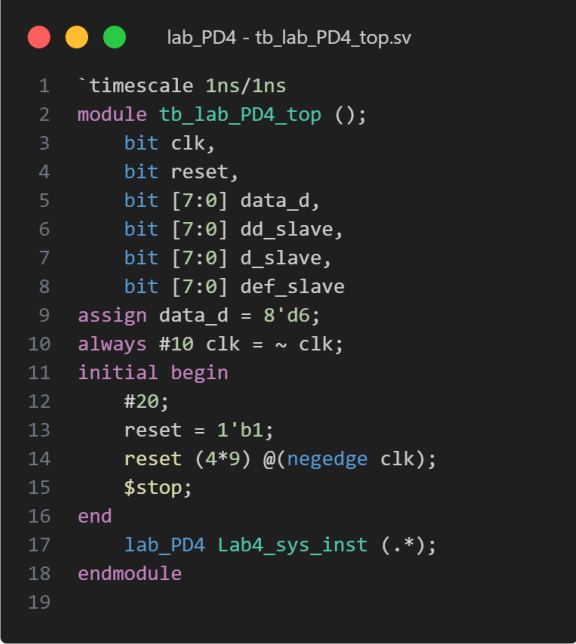
Можем увидеть, что полученная в RTL Viewer схема совпадает с той, что была задана по условию (в зелёном блоке отображается тот фрагмент системы, который был создан средствами PD).

3. Тестирование проекта

3.1. Тестирование средствами ModelSim

Создание тестового файла

Добавим тест первого класса для созданного проекта:



```
lab_PD4 - tb_lab_PD4_top.sv

1  `timescale 1ns/1ns
2  module tb_lab_PD4_top ();
3      bit clk,
4      bit reset,
5      bit [7:0] data_d,
6      bit [7:0] dd_slave,
7      bit [7:0] d_slave,
8      bit [7:0] def_slave
9  assign data_d = 8'd6;
10 always #10 clk = ~ clk;
11 initial begin
12     #20;
13     reset = 1'b1;
14     reset (4*9) @(negedge clk);
15     $stop;
16 end
17     lab_PD4 Lab4_sys_inst (.*);
18 endmodule
19
```

Рис. 38 – Тестовый файл tb_lab_PD4_top.sv

Укажем созданный файл в качестве основного тестового файла, который будет выполняться при симуляции средствами ModelSim:

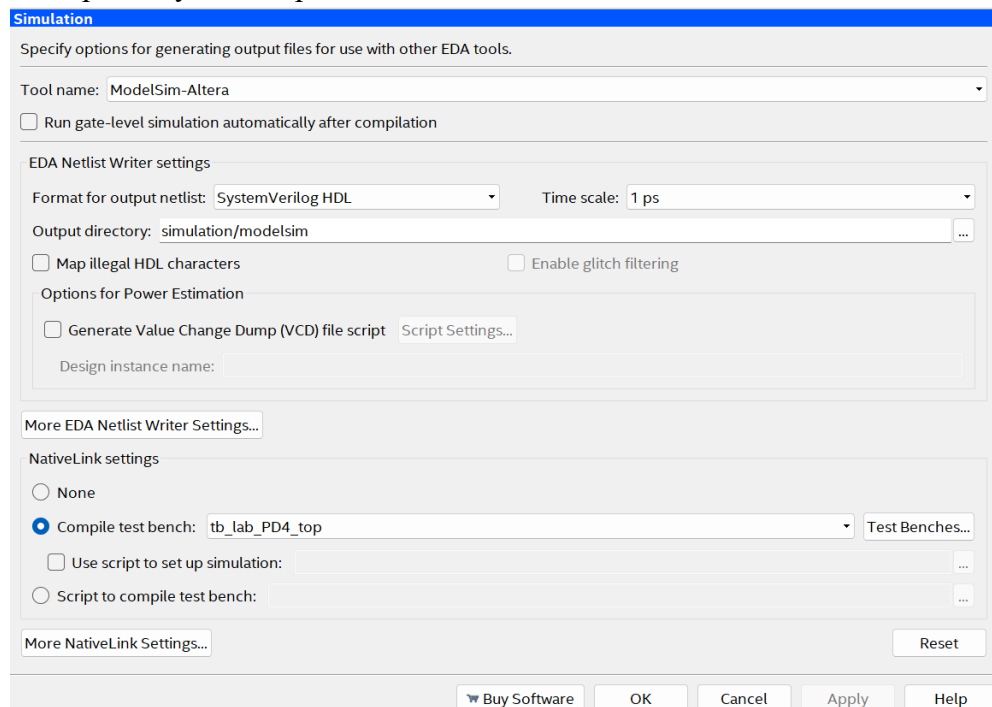


Рис. 39 – Тестовый файл tb_lab_PD2_top.sv

Симуляция средствами ModelSim

Выполним компиляцию проекта средствами ModelSim. Для этого запустим wave.do файл:

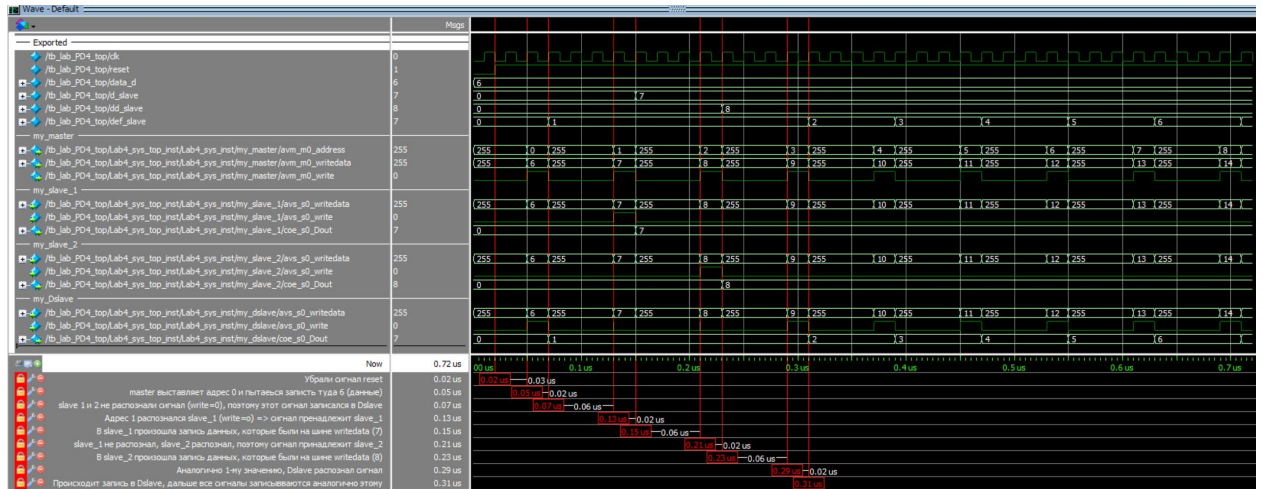


Рис. 40 – Моделирование проекта средствами ModelSim

Изначально сигнал reset был равен 0, сделали его = 1, устройство начало работать.

master выставляет адрес 0 и пытается записать туда 6 (это данные, которые находятся на шине writedata).

1) Адрес 0:

slave_1 и **slave_2** не смогли распознать сигнал (мы это поняли так как в соответствующей им строке write значение сигнала по прежнему = 0) => сигнал теперь принадлежит (будет записан в) **Dslave** (в этот slave всегда происходит запись, когда остальные не смогли распознать сигналы, т. к. данные передаются прямо с шины, перед этим проходя через slave_1 и slave_2). Заметим, что после того, как произошла запись Dout принимает значение = 1. Это связано с тем, что нам не интересно, что записывается в **Dslave**, поэтому на выход просто поставлен счётчик, который увеличивается каждый раз, когда мы что-то выводит на этот slave.

2) Адрес 1:

Адрес 1 был распознан **slave_1** (write = 1) => сигнал теперь принадлежит **slave_1**. При этом до **slave_2** и **Dslave** уже этот сигнал не дойдёт (сигнал идёт в порядке **slave_1** → **slave_2** → **Dslave** и, если распознаётся каким-то из сигналов, то дальше не идёт).

3) Адрес 2:

Адрес 2 не был распознан **slave_1**, но был распознан **slave_2** (write = 1), происходит запись в **slave_2**

4) Адрес 3:

Адрес 3, как и первый, не был распознан **slave_1** и **slave_2**, происходит запись в **Dslave** значения счётчика + 1.

5) Оставшиеся от 255 значения, которые не принадлежат, ни **slave_1**, ни **slave_2**, отправляются на **Dslave**.

*За тем, какие данные куда и когда записались удобно следить последним 4 сигналам блока Esported (верхний блок перед my_master).

Изменение тестового файла

Изменим тест таким образом, чтобы он обеспечивал генерацию числа тактовых сигналов, необходимых для формирования master'ом шины Avalon-MM 260 сигналов, для этого изменим тестовый файл следующим образом (на 14-й строке вместо 9 поставим 260):

```
lab_PD4 - tb_lab_PD4_top.v  
  
1 `timescale 1ns/1ns  
2 module tb_lab_PD4_top ();  
3     bit clk;  
4     bit reset;  
5     bit [7:0] data_d;  
6     bit [7:0] dd_slave;  
7     bit [7:0] d_slave;  
8     bit [7:0] def_slave;  
9     assign data_d = 8'd6;  
10    always #10 clk = ~ clk;  
11    initial begin  
12        #20;  
13        reset = 1'b1;  
14        repeat (4*260) @(negedge clk);  
15        $stop;  
16    end  
17    lab_PD4_top Lab4_sys_top_inst (.*);  
18 endmodule  
19
```

Рис. 41 – Изменённый тестовый файл

Симуляция средствами ModelSim (все значения)

Получившаяся wave будет во много раз больше, т. к. в ней отображается больше значений, чтобы было наглядно рассмотрим только последнюю её часть, которая представлена на ниже:

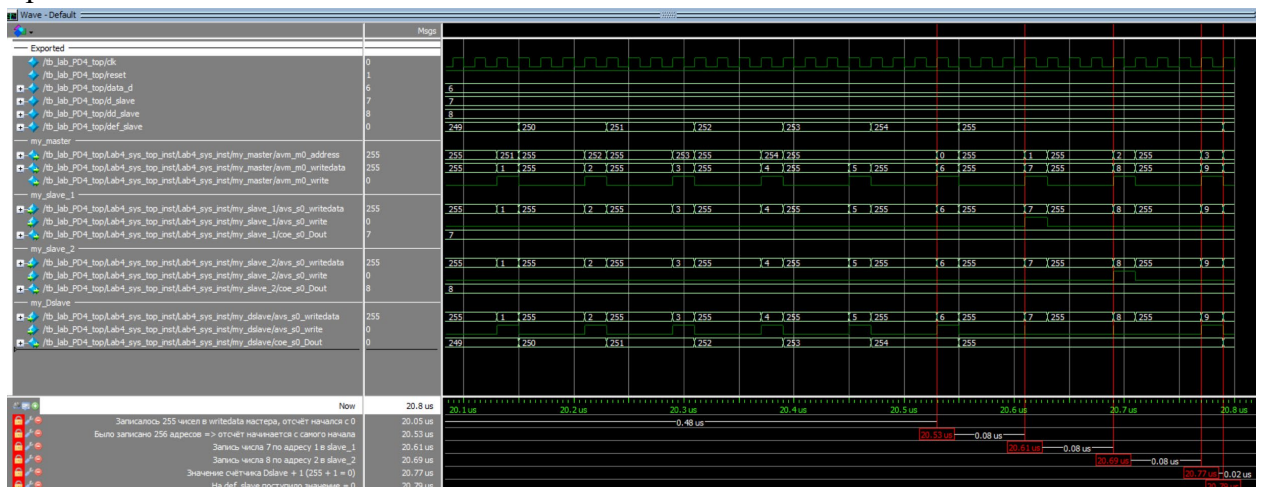


Рис. 42 – Моделирование проекта средствами ModelSim (все значения)

Заметим, что временная диаграмма работает корректно. После записи 255 чисел в writedata мастера, значения пошли снова от 0 до 255. К моменту, когда значение стало равно 6, адрес сбросился и тоже начал идти с 0. Таким образом, цикл завершился и дальше происходит запись чисел аналогично тому, как показано на Рис. 40. Таким образом, число 7 по адресу 1 записалось в slave_1, число 8 по адресу 2 записалось в slave_2 и т. д.

Заметим, что за всё это время к модулю Dslave было $256 + 1$ обращений = 257. Здесь 256 — это предпоследнее значение в `coe_s0_Dout`, которое увеличивалось на 1 при каждом новом обращении к Dslave, а + 1, т. к. в самом конце `coe_s0_Dout` обнулится из-за ещё одного обращения.

3.2. Тестирование средствами Signal Tap II

Создание файла для отладки

Создадим файл db_lab_PD4_top.sv для отладки модуля lab_PD4_top:

```
lab_PD4 - db_lab_PD4_top.sv
1 `timescale 1ns/1ns
2 module db_lab_PD4_top (
3     (* altera_attribute = "-name IO_STANDARD \"3.3-V LVCMS\" \"\", chip_pin = \"23\" *)
4     input bit clk
5 );
6     bit reset;
7     bit [7:0] data_d;
8     bit [7:0] dd_slave;
9     bit [7:0] d_slave;
10    bit [7:0] def_slave;
11    SP_unit SP_unit_inst (
12        .source ( {reset, data_d} ),
13        .source_clk (clk)
14    );
15    lab_PD4_top lab_PD4_top_inst (.*)
16 endmodule
17
```

Рис. 43 – Файл для отладки модуля верхнего уровня

Создадим модуль ISSPE, укажем файл db_lab_PD4_top.sv файлом верхнего уровня и убедимся в том, что схема, получаемая в результате компиляции, будет верной:

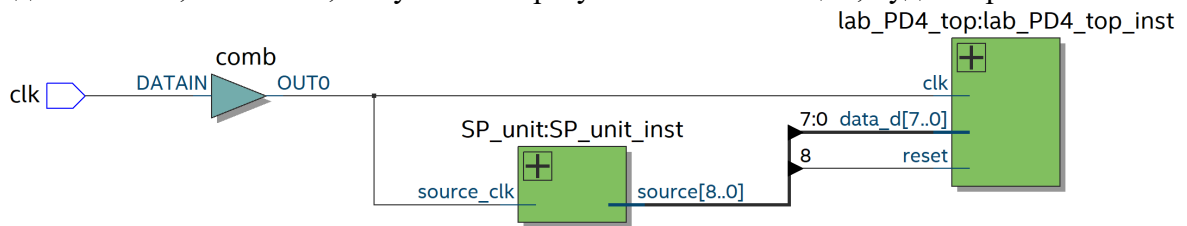


Рис. 44 – Схема проекта с добавлением SP_unit в RTL Viewer

Как видно из схемы SP_unit добавлен корректно.

Выберем сигналы для логического анализатора:

Node Finder

Named: List

Options

Filter: Customize...

Look in: ... ☒ Include subtentities ☒ Hierarchy view

Matching Nodes:

Name	Assignments
db_lab_PD3_top	
clk	Pin_23
lab_PD3_top:Lab3_top_inst	
clk	Unassigned
reset	Unassigned
lab_PD3:lab3_1_inst	
clk_clk	Unassigned
reset_reset_n	Unassigned
> lab_PD3_mm_interconnect_0:mm_interconnect_0	
> my_master:my_master	
> my_slave:my_slave	
> my_slaveWS:my_slavews	
> dout_a_export	Unassigned
> dout_b_export	Unassigned
> ledA	Unassigned
> ledB	Unassigned
> SP_unit:u0	

Nodes Found:

Name	Assignments
lab_PD3_top:Lab3_top_inst reset	Unassigned
lab_PD3_top:Lab3_top_inst ledA	Unassigned
lab_PD3_top:Lab3_top_inst ledB	Unassigned
lab_PD3_top:Lab3_top_inst ...my_master avm_m0_address	Unassigned
lab_PD3_top:Lab3_top_inst ...my_master avm_m0_writedata	Unassigned
lab_PD3_top:Lab3_top_inst ...y_master avm_m0_waitrequest	Unassigned
lab_PD3_top:Lab3_top_inst ...ster:my_master avm_m0_write	Unassigned
lab_PD3_top:Lab3_top_inst la...ve:my_slave avs_s0_writedata	Unassigned
lab_PD3_top:Lab3_top_inst la...slave:my_slave avs_s0_write	Unassigned
lab_PD3_top:Lab3_top_inst la...my_slave avs_s0_waitrequest	Unassigned
lab_PD3_top:Lab3_top_inst la...slave:my_slave coe_s0_Dout	Unassigned
lab_PD3_top:Lab3_top_inst la...my_slavews avs_s0_writedata	Unassigned
lab_PD3_top:Lab3_top_inst la...eWS:my_slavews avs_s0_write	Unassigned
lab_PD3_top:Lab3_top_inst la...y_slavews avs_s0_waitrequest	Unassigned
lab_PD3_top:Lab3_top_inst ...eWS:my_slavews coe_s0_Dout	Unassigned


Insert Close

Настроим окно Signal Tap II так, как показано на рисунке ниже:

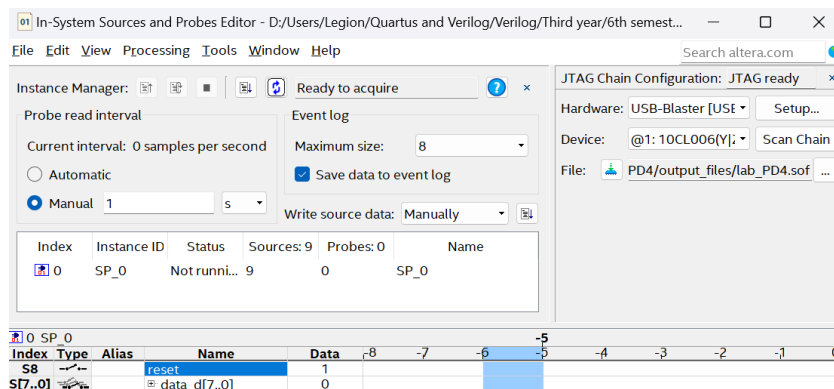
auto_signaltap_0		Node		Lock mode: Allow all changes	Signal Configuration:	
Type	Alias	Name	Data Enable	Trigger Enable	Trigger Conditions	
		lab PD4 toplab PD4 top instlreset	101	101		Clock: clk
		^ lab PD4 toplab PD4 top instl\data df[7..0]			XXXXXXXXXXb	Data
		^ lab PD4 toplab PD4 top instl\dd slave[7..0]			XXXXXXXXXXb	Sample depth: 64 RAM type: Auto
		^ lab PD4 toplab PD4 top instl\dd slave[7..0]			XXXXXXXXXXb	<input type="checkbox"/> Segmented: 2 32 sample segments
		^ lab PD4 toplab PD4 top instl\dd slave[7..0]			XXXXXXXXXXb	<input type="checkbox"/> Nodes Allocated: <input checked="" type="radio"/> Auto <input type="radio"/> Manual: 101
		^ lab PD4 toplab PD4 top instl\lab PD4Lab4 sys instl\m master\m m0 address[7..0]			XXXXXXXXXXb	Pipeline factor: 0
		^ lab PD4 toplab PD4 top instl\lab PD4Lab4 sys instl\m master\m m0 wdata[7..0]			XXXXXXXXXXb	Storage qualifier:
		^ lab PD4 toplab PD4 top instl\lab PD4Lab4 sys instl\m master\m m0 wdata[7..0]			XXXXXXXXXXb	Type: Continuous
		^ lab PD4 toplab PD4 top instl\lab PD4Lab4 sys instl\m master\m m0 wdata[7..0]			XXXXXXXXXXb	Input port:
		^ lab PD4 toplab PD4 top instl\lab PD4Lab4 sys instl\m master\m m0 wdata[7..0]			XXXXXXXXXXb	Nodes Allocated: <input checked="" type="radio"/> Auto <input type="radio"/> Manual: 101
		^ lab PD4 toplab PD4 top instl\lab PD4Lab4 sys instl\m master\m m0 wdata[7..0]			XXXXXXXXXXb	

Тестирование на плате средствами Signal Tap II

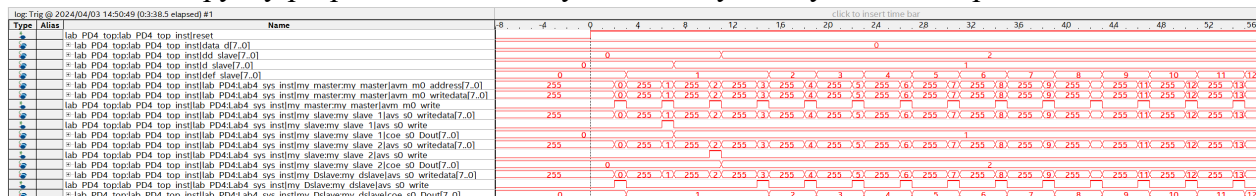
Выполним полную компиляцию. В отчете о компиляции видно, что устройство удовлетворяет временным параметрам.

Slow 1200mV 85C Model Fmax Summary				
 <<Filter>>				
	Fmax	Restricted Fmax	Clock Name	Note
1	93.67 MHz	93.67 MHz	altera reserved tck	

Выполним настройку окна In-System Sources and Probe Editor:



Теперь запустим и выполним проверку корректности работы программы на плате. Выполним загрузку разработанного модуля на плату и запустим тестирование:



Полученная временная диаграмма совпадает с той, что была получена в ходе тестирования проекта средствами ModelSim (Рис. 40 – Рис. 42). Данные поступают и передаются на приёмник корректно.

4. Вывод

В ходе лабораторной работы было создано устройство, включающее мастер Avalon MM и три слейва - два модуля `my_slave` и один модуль `my_Dslave`. Мастер осуществляет адресный доступ к слейвам и управляет передачей данных. Слейвы принимают данные от мастера и, в зависимости от адреса, либо сохраняют их, либо передают дальше.

Преимущество такого устройства заключается в его гибкости и расширяемости. Мастер может направлять данные на различные устройства в зависимости от их адреса, а слейвы могут быть легко добавлены или удалены по мере необходимости. Это делает систему универсальной и адаптивной к различным потребностям.

Такие устройства широко используются во встраиваемых системах, где требуется управление и обмен данными между различными компонентами. Примерами могут служить устройства для управления промышленным оборудованием, автомобильными системами, сетевыми устройствами и другими приложениями, где требуется эффективная передача данных и управление периферийными устройствами.