

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

Отчёт по лабораторной работе Lab_PD3

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

1. Оглавление

1.	Оглавление	2
2.	Задание.....	5
3.	Ход работы	6
3.1.	Создание проекта.....	6
	Подготовка проекта	6
	Начало работы в PD	6
3.2.	Настройка сигналов.....	8
	Настройка clk.....	8
	Настройка sc_fifo_0.....	8
	Настройка MyST_source_0 и MyST_sink_0	8
3.3.	Подключение сигналов	9
	Подключение тактового сигнала	9
	Подключение сигнала Reset.....	9
3.4.	Подключение Avalon-MM интерфейсов	10
3.5.	Экспорт выводов.....	11
3.6.	Настройка my_master	11
3.7.	Анализ системы	13
	Проверка блока.....	13
	Анализ с помощью Schematic	14
	Генерация системы	14
	Анализ подключенных файлов	15
3.8.	Подключение файлов к проекту.....	17
4.	Тестирование проекта	19
4.1.	Тестирование средствами ModelSim	19
	Создание тестового файла.....	19
	Симуляция средствами ModelSim	20
4.2.	Тестирование средствами Signal Tap II	21
	Создание файла для отладки.....	21
	Настройка Signal Tap II.....	21
	Тестирование на плате средствами Signal Tap II	22
5.	Дополнительное задание.....	23
5.1.	Изменение значений по варианту	23
	Изменение адресов slave.....	23
	Изменение компонента master	23
5.2.	Тестирование средствами ModelSim	24
	Симуляция средствами ModelSim	24
5.3.	Тестирование средствами Signal Tap II	24
	Тестирование на плате средствами Signal Tap II	24
6.	Вывод.....	26

Список иллюстраций

Рис. 1 – Структура проекта	5
Рис. 2 – Задания пути к библиотеке IP	6
Рис. 3 – Исходное окно PD	6
Рис. 4 – Добавление компонентов	7
Рис. 5 – Переименование компонентов	7
Рис. 6 – Настройка компонента clk	8
Рис. 7 – Настройка компонента sc_fifo	8
Рис. 8 – Система после переименования компонентов	9
Рис. 9 – Подключение тактового сигнала (1)	9
Рис. 10 – Подключение тактового сигнала (2)	9
Рис. 11 – Подключение сигнала Reset	10
Рис. 12 – Подключение Avalon-ММ интерфейсов	10
Рис. 13 – Фиксация адресов	10
Рис. 14 – Назначение правильных адресов для компонентов	11
Рис. 15 – Проверка корректности адресов	11
Рис. 16 – Экспорт выводов	11
Рис. 17 – Настройка компонента my_master	12
Рис. 18 – Проверка системы	12
Рис. 19 – Символ системы	13
Рис. 20 – Анализ проблемных подключений	13
Рис. 21 – Show System with QSYS Interconnect	14
Рис. 22 – Schematic	14
Рис. 23 – Предустановки окна Generation	15
Рис. 24 – Проверка успешности генерации HDL	15
Рис. 25 – Файл компонента my_master (1)	15
Рис. 26 – Файл компонента my_master (2)	16
Рис. 27 – Файл компонента my_slave	16
Рис. 28 – Файл компонента my_slave	17
Рис. 29 – Подключение файлов к проекту	17
Рис. 30 – Синтаксис файла lab_PD3_top.sv	18
Рис. 31 – Схема проекта в RTL Viewer	18
Рис. 32 – Тестовый файл tb_lab_PD2_top.sv	19
Рис. 33 – Тестовый файл tb_lab_PD2_top.sv	19
Рис. 34 – Моделирование проекта средствами ModelSim	20
Рис. 35 – Файл для отладки модуля верхнего уровня	21
Рис. 36 – Схема проекта с добавлением SP_unit в RTL Viewer	21
Рис. 37 – Сигналы логического анализатора	21
Рис. 38 – Конечный автомат	22
Рис. 39 – Настройка окна Signal Tap II	22
Рис. 40 – Временные характеристики устройства	22
Рис. 41 – Результат SignalTap II	22
Рис. 42 – Изменение адресов для slave'ов	23

Рис. 43 – Изменённое окно Address Map	23
Рис. 44 – Изменённый модуль master.....	23
Рис. 34 – Моделирование проекта средствами ModelSim.....	24
Рис. 51 – Временные характеристики устройства	24
Рис. 52 – Результат SignalTap II.....	25

2. Задание

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

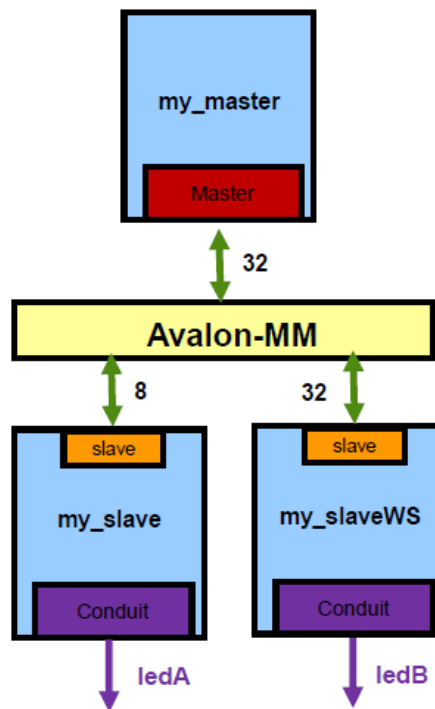


Рис. 1 – Структура проекта

Один из slave'ов – 32-битный, другой – 8-битный → понадобится динамическое выравнивание адресов.

Структура проекта выглядит следующим образом:

- Ведущий осуществляет запись словами по 32 бита
- Ведомый my_slave–8 разрядный. Один цикл записи 32 разрядного слова от Ведущего будет преобразован системой соединений в 4 цикла записи 8 разрядными словами (на время этих четырёх циклов Ведущий будет приостановлен – он получит сигнал waiterquest от системы соединений).
- Ведомый my_slaweWS–32 разрядный. Он, по получению от Ведущего сигнала write,выставляет (на один период тактового сигнала) сигнал waitrequest–приостанавливает Ведущего на один период тактового сигнала. Затем осуществляет запись данных.

3. Ход работы

3.1. Создание проекта

Подготовка проекта

Создадим проект и зададим путь к библиотеке IP:

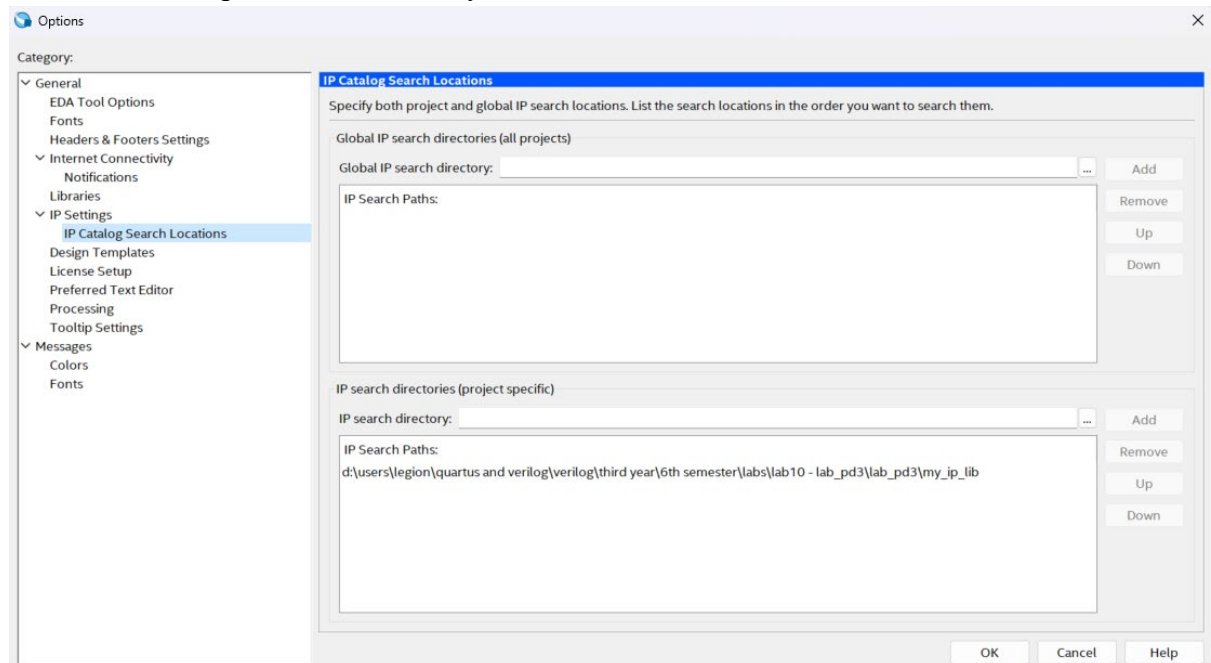


Рис. 2 – Задания пути к библиотеке IP

Начало работы в PD

Откроем PD и сохраним систему:

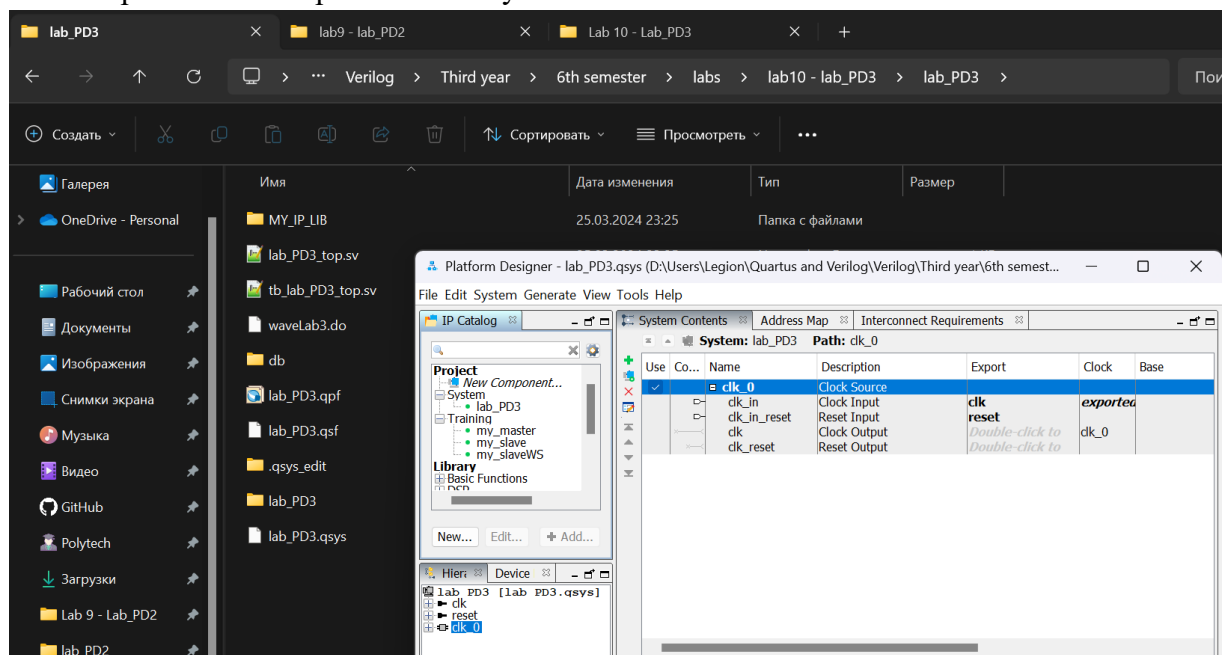


Рис. 3 – Исходное окно PD

Добавим компоненты: my_master, my_slave, my_slaveWS. Таким образом, получим следующую картинку (в окне Hierarchy слева отображаются все добавленные компоненты):

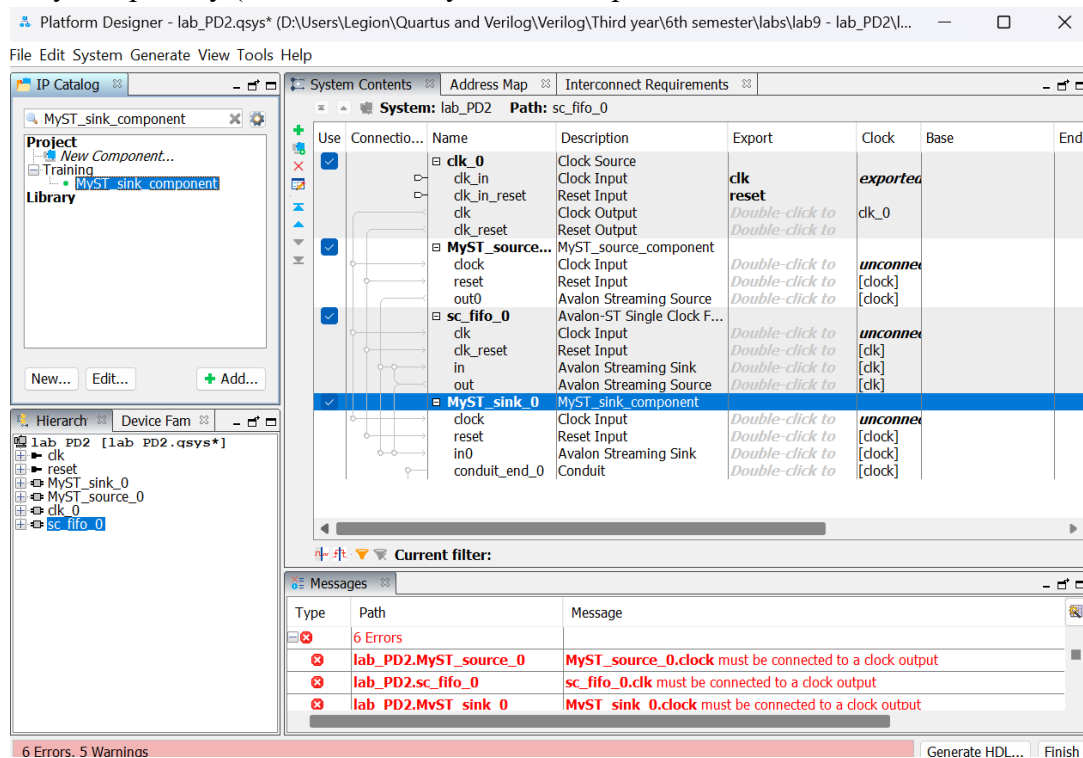


Рис. 4 – Добавление компонентов

Наличие ошибок связано с тем, что настройка модулей не производилась, т. к. она будет рассмотрена дальше.

Переименуем компоненты (уберём постфикс `_0`):

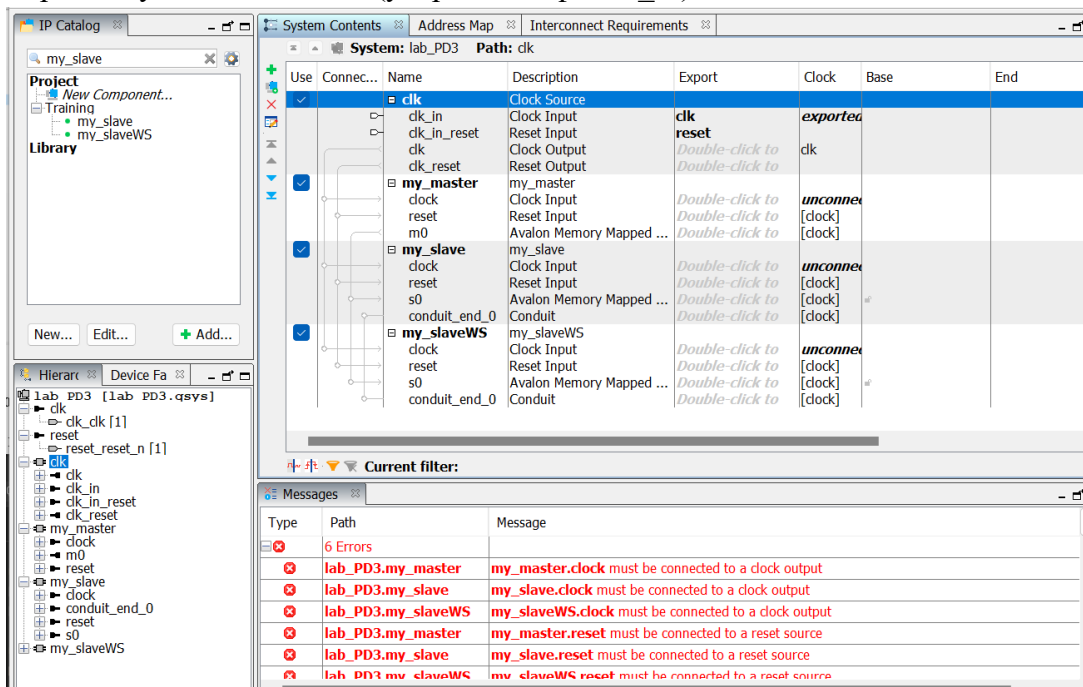


Рис. 5 – Переименование компонентов

3.2. Настройка сигналов

Настройка clk

Зададим значение Reset synchronous edges = Deassert

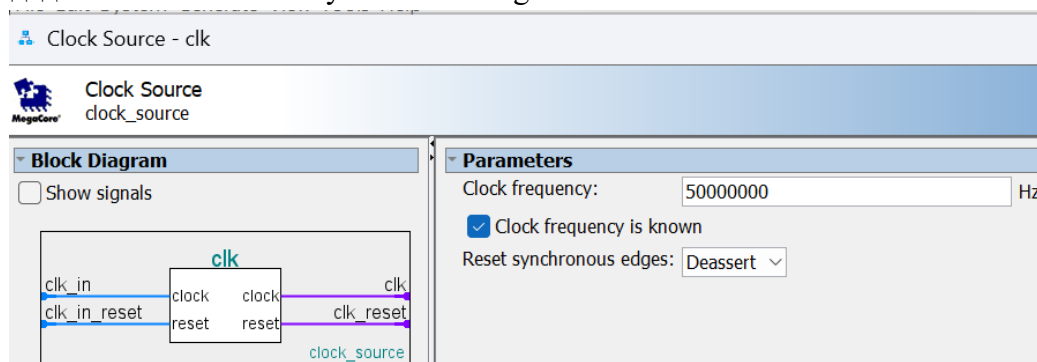


Рис. 6 – Настройка компонента clk

Настройка sc_fifo_0

Зададим значение Bits per symbol = 4:

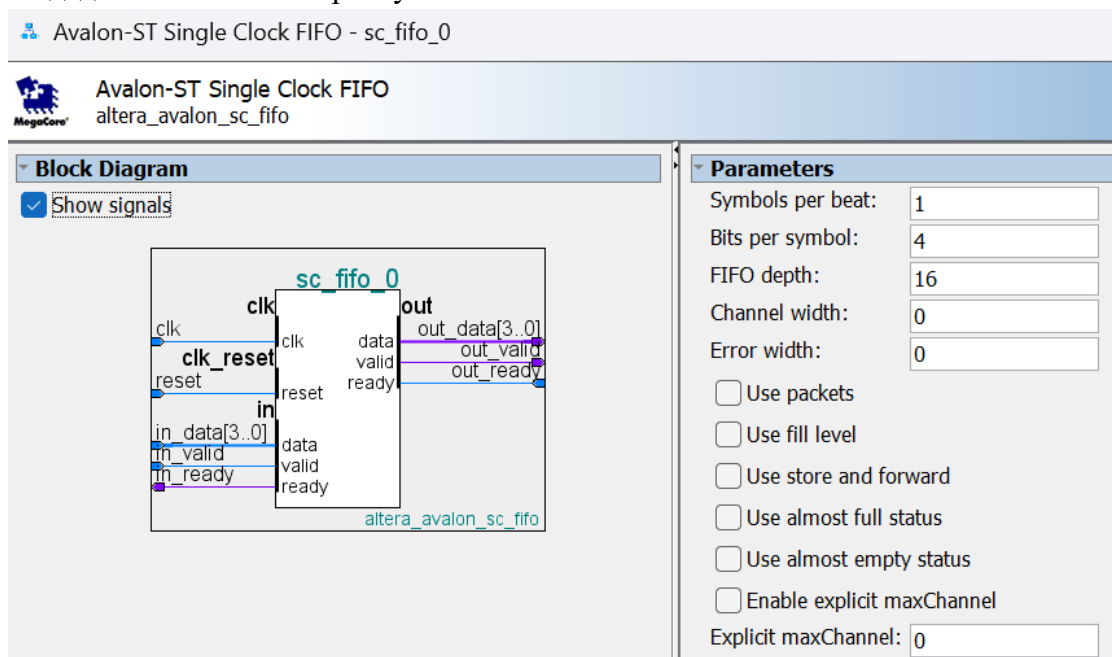


Рис. 7 – Настройка компонента sc_fifo

Запись данных будет происходить следующим образом: 100 – счёт на сложение, 200 – счёт на вычитание.

Настройка MyST_source_0 и MyST_sink_0

Переименуем MyST_source_0 и MyST_sink_0 в MyST_source и MyST_sink соответственно. Получившаяся структура будет выглядеть следующим образом:

Use	Connectio...	Name	Description	Export	Clock	Base
<input checked="" type="checkbox"/>		clk	Clock Source			
		clk_in	Clock Input	clk	exported	
		clk_in_reset	Reset Input	reset	[clk_in]	
		clk_reset	Clock Output	clk	clk	
		clk_reset	Reset Output	reset		
<input checked="" type="checkbox"/>		MyST_source	MyST_source_component			
		clock	Clock Input	Double-click to	unconnex	
		reset	Reset Input	Double-click to	[clock]	
		out0	Avalon Streaming Source	Double-click to	[clock]	
<input checked="" type="checkbox"/>		sc_fifo	Avalon-ST Single Clock F...			
		clk	Clock Input	Double-click to	unconnex	
		clk_reset	Reset Input	Double-click to	[clk]	
		in	Avalon Streaming Sink	Double-click to	[clk]	
		out	Avalon Streaming Source	Double-click to	[clk]	
<input checked="" type="checkbox"/>		MyST_sink	MyST_sink_component			
		clock	Clock Input	Double-click to	unconnex	
		reset	Reset Input	Double-click to	[clock]	
		in0	Avalon Streaming Sink	Double-click to	[clock]	
		conduit end 0	Conduit	Double-click to	MyST sink reset	

Рис. 8 – Система после переименования компонентов

3.3. Подключение сигналов

Подключение тактового сигнала

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

Connected	Connection	Clock Crossing	Data Width
<input checked="" type="checkbox"/>	clk.clk/my_master.clock		
<input checked="" type="checkbox"/>	clk.clk/my_slave.clock		
<input checked="" type="checkbox"/>	clk.clk/my_slaveWS.clock		

Рис. 9 – Подключение тактового сигнала (1)

Выполним Filter → Clock and Reset Interfaces и убедимся, что соединения выполнены корректно:

Use	Co...	Name	Description	Export	Clock	Base
<input checked="" type="checkbox"/>		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	
<input checked="" type="checkbox"/>		my_master	my_master			
		clock	Clock Input	Double-click to	clk	
		reset	Reset Input	Double-click to	[clock]	
<input checked="" type="checkbox"/>		my_slave	my_slave			
		clock	Clock Input	Double-click to	clk	
		reset	Reset Input	Double-click to	[clock]	
<input checked="" type="checkbox"/>		my_slaveWS	my_slaveWS			
		clock	Clock Input	Double-click to	clk	
		reset	Reset Input	Double-click to	[clock]	

Рис. 10 – Подключение тактового сигнала (2)

Подключение сигнала Reset

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

System: lab_PD3 Path: clk.clk_reset						
Use	Co...	Name	Description	Export	Clock	Base
<input checked="" type="checkbox"/>		clk	Clock Source			
		clk_in	Clock Input	clk	exported	
		clk_in_reset	Reset Input	reset	[clk_in]	
		clk	Clock Output	clk	clk	
<input checked="" type="checkbox"/>		clk_reset	Reset Output	clk	clk	
<input checked="" type="checkbox"/>		my_master	my_master			
		clock	Clock Input	clk	clk	
		reset	Reset Input	reset	[clock]	
<input checked="" type="checkbox"/>		my_slave	my_slave			
		clock	Clock Input	clk	clk	
		reset	Reset Input	reset	[clock]	
<input checked="" type="checkbox"/>		my_slaveWS	my_slaveWS			
		clock	Clock Input	clk	clk	
		reset	Reset Input	reset	[clock]	

Рис. 11 – Подключение сигнала Reset

3.4. Подключение Avalon-MM интерфейсов

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

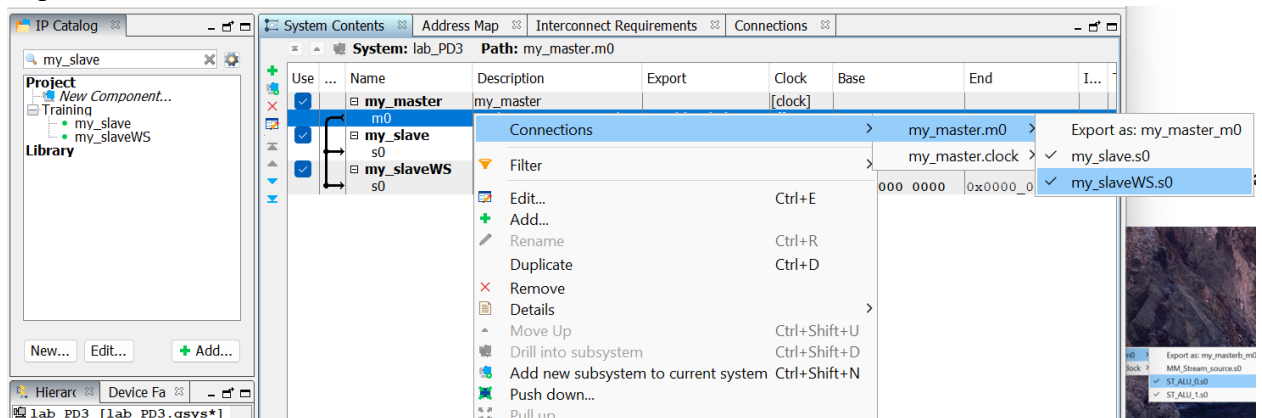


Рис. 12 – Подключение Avalon-MM интерфейсов

Заметим, что в окне Msgs высвечивается ошибка, указывающая на то, что базовые адреса заданы неправильно. Поменяем адреса, проверим, что они верны, и зафиксируем их (нажать символ замка у каждого из адресов):

System: lab_PD1_sys Path: my_masterA.m0								
Use	Co...	Name	Description	Export	Clock	Base	End	I...
<input checked="" type="checkbox"/>		my_masterA	my_masterA_component		[clock]			
<input checked="" type="checkbox"/>		m0	Avalon Memory Mapped Master	clk	clk			
<input checked="" type="checkbox"/>		MM_Stream_source	MM_Stream_source_component	clk	clk	0x0000_0000	0x0000_0003	
<input checked="" type="checkbox"/>		s0	Avalon Memory Mapped Slave	clk	clk			
<input checked="" type="checkbox"/>		my_masterB	my_masterB_component		[clock]			
<input checked="" type="checkbox"/>		m0	Avalon Memory Mapped Master	clk	clk			
<input checked="" type="checkbox"/>		ST_ALU_0	ST_ALU_component	clk	clk	0x0000_0004	0x0000_0007	
<input checked="" type="checkbox"/>		s0	Avalon Memory Mapped Slave	clk	clk			
<input checked="" type="checkbox"/>		ST_ALU_1	ST_ALU_component	clk	clk	0x0000_0000	0x0000_0003	
<input checked="" type="checkbox"/>		s0	Avalon Memory Mapped Slave	clk	clk			

Рис. 13 – Фиксация адресов

Откроем окно закладок View → Address Map. В окне появятся Ведущие (master) и ведомые (slave) шины, т. е. столбцы и строки Avalon MM соответственно. Проверим, что они выведены корректно. Для этого компоненту my_slave.s0 назначим базовый адрес = 0 и выполним System → Assign Base Address:

Use	Name	Description	Export	Clock	Base	End	I...
<input checked="" type="checkbox"/>	my_master m0	my_master Avalon Memory Mapped ...	Double-click to	[clock]			
<input checked="" type="checkbox"/>	my_slave s0	my_slave Avalon Memory Mapped ...	Double-click to	clk	0x0000_0000	0x0000_0000	
<input checked="" type="checkbox"/>	my_slaveWS s0	my_slaveWS Avalon Memory Mapped ...	Double-click to	clk	0x0000_0004	0x0000_0007	

Рис. 14 – Назначение правильных адресов для компонентов

Вкладка Address Map будет выглядеть следующим образом:

System Contents	Address Map	Interconnect Requirements
System: lab_PD3 Path: my_slave.s0		
	my_master.m0	
my_slave.s0	0x0000_0000 - 0x0000_0000	
my_slaveWS.s0	0x0000_0004 - 0x0000_0007	

Рис. 15 – Проверка корректности адресов

Запомним базовые адреса:

- my_slave.s0 = 0,
- my_slaveWS.s0 = 0.

*Позже он будут указываться при настройке модулей my_slave и my_slaveWS соответственно

3.5. Экспорт выводов

Проведём экспорт выводов путём задания имён для выделенных модулей в столбце Export (значение dout_a и dout_b в соответствующем столбце выделенных строк):

Use	Name	Description	Export	Clock	Base	End	I...
<input checked="" type="checkbox"/>	my_slave clock reset s0	my_slave Clock Input Reset Input Avalon Memory Mapped ...	Double-click to Double-click to Double-click to	clk [clock] [clock]	0x0000_0000	0x0000_0000	
<input checked="" type="checkbox"/>	my_slaveWS clock reset s0	my_slaveWS Clock Input Reset Input Avalon Memory Mapped ...	Double-click to Double-click to Double-click to	clk [clock] [clock]	0x0000_0004	0x0000_0007	

Рис. 16 – Экспорт выводов

3.6. Настройка my_master

Зададим следующие значения:

- address_1 = 0
- data_2 = 287454020, это соответствует 32'h11223344, что будет удобно при запуске моделирования и последующем рассмотрении результатов на временной диаграмме
- address_2 = 4
- data_2 = 61166, что соответствует 32'h0000eeee (принцип задания такого числа аналогичен data_1, для удобства анализа передачи: big или little endian)

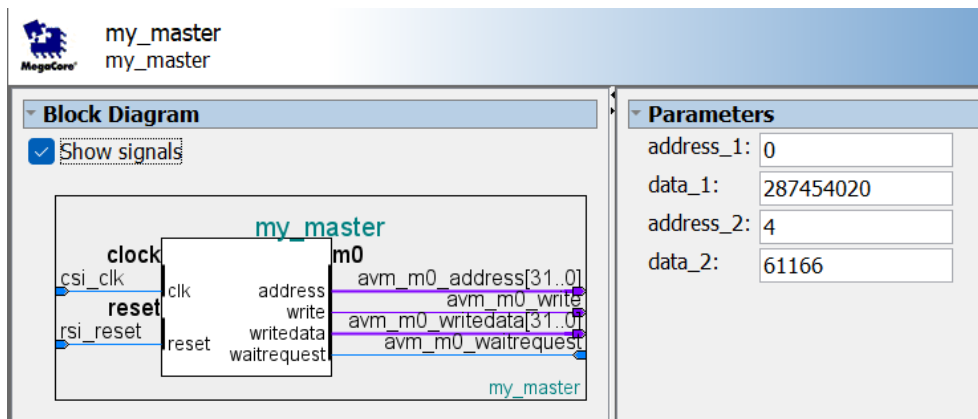


Рис. 17 – Настройка компонента `my_master`

Заметим, что значения `address_1` и `address_2` соответствуют значениям slave'ов.

Убедимся в том, что система выглядит корректно и в поле Messages нет ошибок:

The screenshot shows the Platform Designer interface for a system named `lab_PD3`. The **System Contents** table lists the following components and their connections:

Use	Connec...	Name	Description	Export	Clock	Base	End
✓		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			
✓		my_master	my_master				
		clock	Clock Input	Double-click to	clk		
		reset	Reset Input	Double-click to	[dclk]		
		m0	Avalon Memory Mapped ...	Double-click to	[dclk]		
✓		my_slave	my_slave				
		clock	Clock Input	Double-click to	clk		
		reset	Reset Input	Double-click to	[dclk]		
		s0	Avalon Memory Mapped ...	Double-click to	[dclk]	0x0000_0000	0x0000_0000
		conduit_end_0	Conduit	Double-click to	dout_a		
✓		my_slaveWS	my_slaveWS				
		clock	Clock Input	Double-click to	clk		
		reset	Reset Input	Double-click to	[dclk]		
		s0	Avalon Memory Mapped ...	Double-click to	[dclk]	0x0000_0004	0x0000_0007
		conduit_end_0	Conduit	Double-click to	dout_b		

The **Messages** panel at the bottom is empty, indicating no errors. The status bar at the bottom shows "0 Errors, 0 Warnings" and buttons for "Generate HDL..." and "Finish".

Рис. 18 – Проверка системы

3.7. Анализ системы

Проверка блока

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

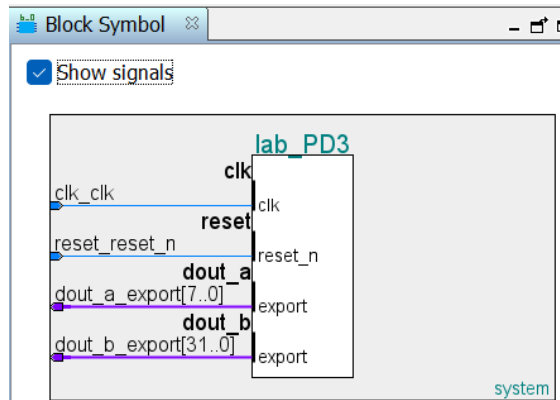


Рис. 19 – Символ системы

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

Use	Connec...	Name	Description	Export	Clock	Base	End	I
<input checked="" type="checkbox"/>		clk	Clock Source					
<input checked="" type="checkbox"/>		clk_in	Clock Input	clk	exported			
<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	reset	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 ...	dout_a	[clock]			
<input checked="" type="checkbox"/>		my_slave	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 ...	dout_b	[clock]			
<input checked="" type="checkbox"/>		conduit_end_0	Conduit			0x0000_0000	0x0000_0000	
<input checked="" type="checkbox"/>		my_slaveWS	my_slaveWS					
<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 ...	dout_b	[clock]			
<input checked="" type="checkbox"/>		conduit_end_0	Conduit			0x0000_0004	0x0000_0007	

Рис. 20 – Анализ проблемных подключений

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

System Contents Parameters Memory-Mapped Interconnect System: lab_PD3								
Use	Connections	Name	Description	Export	Clock	Base	End	I
<input checked="" type="checkbox"/>		mm_intercon...	MM Interconnect					
<input checked="" type="checkbox"/>		clk_clk	Clock Input	Double-click to	clk			
<input checked="" type="checkbox"/>		my_master_re...	Reset Input	Double-click to	[clk_clk]			
<input checked="" type="checkbox"/>		my_master_m0	Avalon Memory Mapped ...	Double-click to	[clk_clk]	# 0x0000_0000	0xffff_ffff	
<input checked="" type="checkbox"/>		my_slave_s0	Avalon Memory Mapped ...	Double-click to	[clk_clk]			
<input checked="" type="checkbox"/>		my_slaveWS_s0	Avalon Memory Mapped ...	Double-click to	[clk_clk]			
<input checked="" type="checkbox"/>		clk	Clock Source					
<input checked="" type="checkbox"/>		clk_in	Clock Input	clk	exported			
<input checked="" type="checkbox"/>		clk_in_reset	Reset Input	reset	[clk_in]			
<input checked="" type="checkbox"/>		clk	Clock Output	Double-click to	clk			
<input checked="" type="checkbox"/>		clk_reset	Reset Output	Double-click to	clk			
<input checked="" type="checkbox"/>		my_master	my_master					
<input checked="" type="checkbox"/>		clock	Clock Input	Double-click to	clk			
<input checked="" type="checkbox"/>		reset	Reset Input	Double-click to	[clock]			
<input checked="" type="checkbox"/>		m0	Avalon Memory Mapped ...	Double-click to	[clock]			
<input checked="" type="checkbox"/>		my_slave	my_slave					
<input checked="" type="checkbox"/>		clock	Clock Input	Double-click to	clk			
<input checked="" type="checkbox"/>		reset	Reset Input	Double-click to	[clock]			
<input checked="" type="checkbox"/>		s0	Avalon Memory Mapped ...	Double-click to	[clock]	# 0x0000	0x0000	
<input checked="" type="checkbox"/>		conduit_end_0	Conduit	Double-click to	dout_a			
<input checked="" type="checkbox"/>		my_slaveWS	my_slaveWS					
<input checked="" type="checkbox"/>		clock	Clock Input	Double-click to	clk			
<input checked="" type="checkbox"/>		reset	Reset Input	Double-click to	[clock]			
<input checked="" type="checkbox"/>		s0	Avalon Memory Mapped ...	Double-click to	[clock]	# 0x0000	0x0003	
<input checked="" type="checkbox"/>		conduit_end_0	Conduit	Double-click to	dout_b			

Рис. 21 – 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:

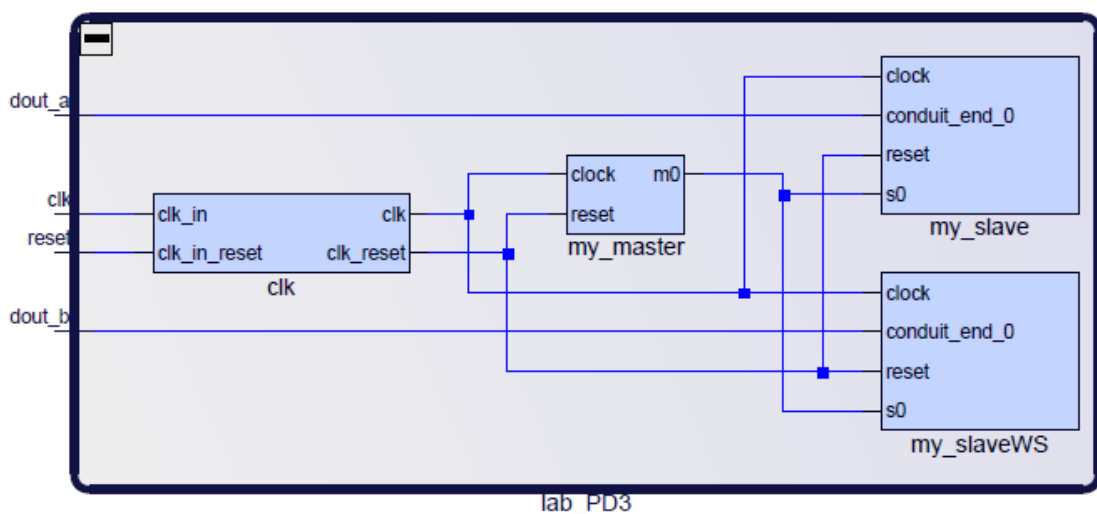


Рис. 22 – Schematic

Генерация системы

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

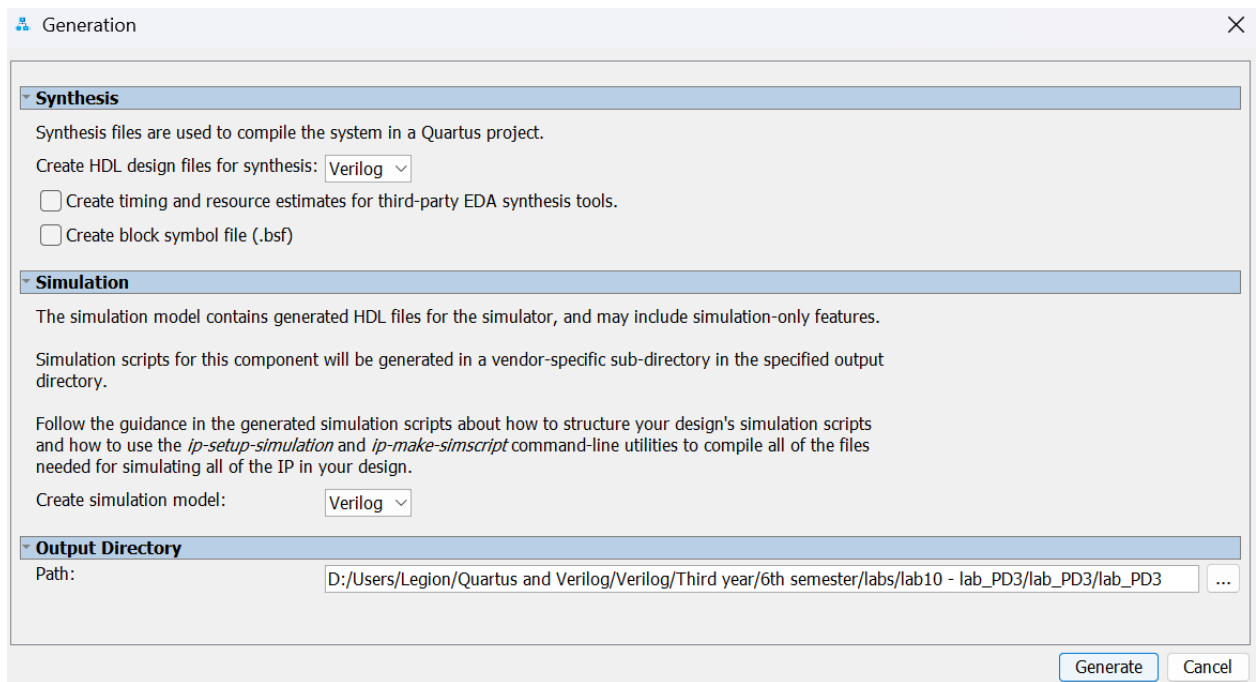


Рис. 23 – Предустановки окна Generation

Удостоверимся в том, что генерация прошла успешно:

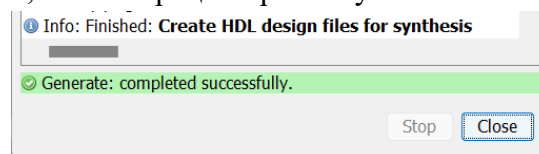


Рис. 24 – Проверка успешности генерации HDL

Анализ подключенных файлов

Таким образом, получили 3 файла: my_master.sv, my_slave.sv и my_slaveSV:



Рис. 25 – Файл компонента my_master (1)

```

lab_PD3 - my_master.sv
15 typedef enum bit[2:0] {initSM, del1, wr1D, del2, wr2D, ended } fsm_type;
16 fsm_type fsm_MM;
17
18 always_ff @ (posedge csi_clk)
19 if (rsi_reset) fsm_MM <= initSM;
20 else
21     case (fsm_MM)
22         initSM : fsm_MM <= del1;
23         del1   : fsm_MM <= wr1D;
24         wr1D   : if (avm_m0_waitrequest) fsm_MM <= wr1D;
25                 else fsm_MM <= del2;
26         del2   : fsm_MM <= wr2D;
27         wr2D   : if (avm_m0_waitrequest) fsm_MM <= wr2D;
28                 else fsm_MM <= ended;
29         ended  : fsm_MM <= ended;
30     endcase
31 always_comb
32 begin
33     case (fsm_MM)
34         wr1D:
35             begin
36                 avm_m0_address    = address_1;
37                 avm_m0_write      = 1'd1;
38                 avm_m0_writedata  = data_1;
39             end
40         wr2D:
41             begin
42                 avm_m0_address    = address_2;
43                 avm_m0_write      = 1'd1;
44                 avm_m0_writedata  = data_2;
45             end
46         default
47             begin
48                 avm_m0_address    = 32'd255;
49                 avm_m0_write      = 1'd0;
50                 avm_m0_writedata  = 32'd255;
51             end
52     endcase
53 end
54 endmodule
55

```

Рис. 26 – Файл компонента my_master (2)

Файл my_maser содержит анализ waitrequest (формируются сигналы либо по параметрам, которые мы задаём, либо значение = 255 в промежутках, которое отвечает за «пустой обмен», значение именно такое, чтобы было нагляднее при моделировании средствами Signal Tap II), интерфейсы clock, reset и master.

```

lab_PD3 - my_slave.sv
1 `timescale 1 ns / 1 ns
2 module my_slave (
3     // clock and reset
4     input bit csi_clk, // clock
5     input bit rsi_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
14 assign avs_s0_waitrequest = 1'b0;
15
16 always_ff @(posedge csi_clk)
17     if (rsi_reset) coe_s0_Dout <= 8'd0;
18     else if (avs_s0_write) coe_s0_Dout <= avs_s0_writedata;
19 endmodule
20

```

Рис. 27 – Файл компонента my_slave

Модуль my_slave (8-битный) – берёт 8-битные данные, у него есть интерфейс MM Slave. Но он НЕ умеет формировать сигнал waitrequest, он занулён. Осуществляет по сигналу write запись тех данных, которые приходят с шины writedata во внутренний регистр.

```
lab_PD3 - my_slaveWS.sv

1 `timescale 1 ns / 1 ns
2 module my_slaveWS (
3   input bit          csi_clk,
4   input bit          rsi_reset,
5   input bit [31:0]   avs_s0_writedata,
6   input bit          avs_s0_write,
7   output bit [31:0]  coe_s0_Dout,
8   output bit         avs_s0_waitrequest);
9
10  bit temp_write;
11
12  always_ff @(posedge csi_clk)
13    if (rsi_reset) temp_write <= '0;
14    else temp_write <= avs_s0_write;
15
16  assign avs_s0_waitrequest = avs_s0_write & ~temp_write;
17
18  always_ff @(posedge csi_clk)
19    if (rsi_reset) coe_s0_Dout <= 32'd0;
20    else if (avs_s0_write) coe_s0_Dout <= avs_s0_writedata;
21 endmodule
22
```

Рис. 28 – Файл компонента my_slave

Модуль my_slaveWS (32-битный) – умеет формировать сигнал waitrequest, получив запрос write, он его записывает в регистр, и только на следующем такте убирает ,этот сигнал, т. е. сделана задержка в 1 такт.

3.8. Подключение файлов к проекту

Подключим файлы к проекту в Quartus

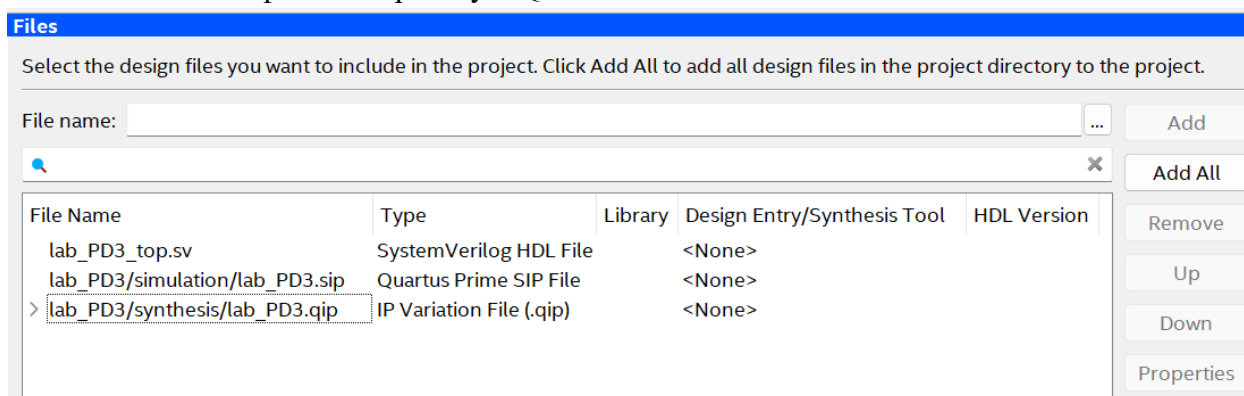


Рис. 29 – Подключение файлов к проекту

Синтаксис файла Lab2_top.sv:

```

lab_PD3 - lab_PD3_top.sv

1 `timescale 1 ns / 1 ns
2 module lab_PD3_top (
3     input bit clk,
4     input bit reset,
5     output bit [7:0] ledA,
6     output bit [31:0] ledB
7 );
8 lab_PD3 lab3_1_inst (
9     .clk_clk      (clk),
10    .reset_reset_n (reset),
11    .dout_a_export (ledA),
12    .dout_b_export (ledB)
13 );
14 endmodule
15

```

Рис. 30 – Синтаксис файла lab_PD3_top.sv

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

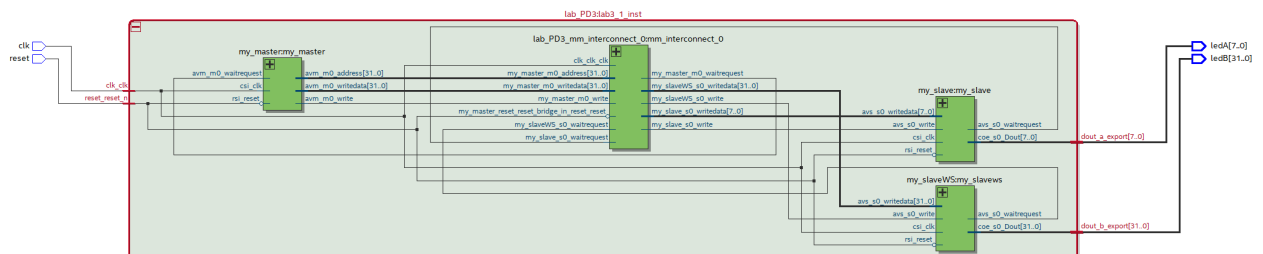


Рис. 31 – Схема проекта в RTL Viewer

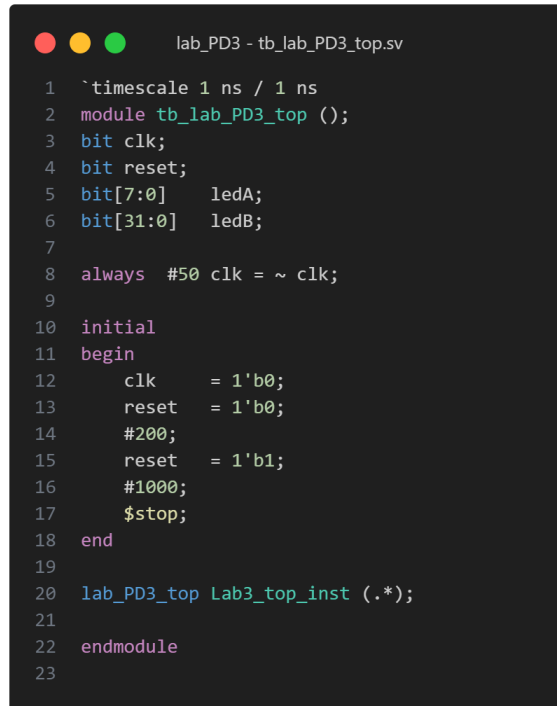
Можем увидеть, что полученная в RTL Viewer схема совпадает с той, что была задана по условию (в зелёном блоке отображается тот фрагмент системы, который был создан средствами PD).

4. Тестирование проекта

4.1. Тестирование средствами ModelSim

Создание тестового файла

Добавим тест первого класса для созданного проекта:



```
lab_PD3 - tb_lab_PD3_top.sv

1 `timescale 1 ns / 1 ns
2 module tb_lab_PD3_top ();
3 bit clk;
4 bit reset;
5 bit[7:0] ledA;
6 bit[31:0] ledB;
7
8 always #50 clk = ~ clk;
9
10 initial
11 begin
12     clk      = 1'b0;
13     reset    = 1'b0;
14     #200;
15     reset    = 1'b1;
16     #1000;
17     $stop;
18 end
19
20 lab_PD3_top Lab3_top_inst (.*);
21
22 endmodule
23
```

Рис. 32 – Тестовый файл tb_lab_PD2_top.sv

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

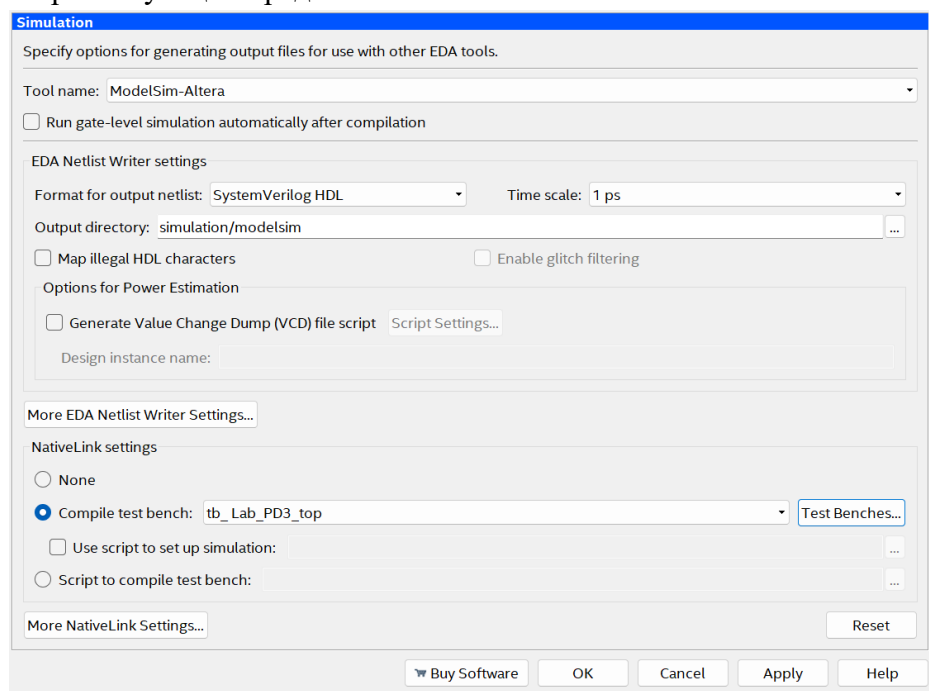


Рис. 33 – Тестовый файл tb_lab_PD2_top.sv

Симуляция средствами ModelSim

Выполним компиляцию проекта средствами ModelSim. Для этого запустим waveLab2.do файл:

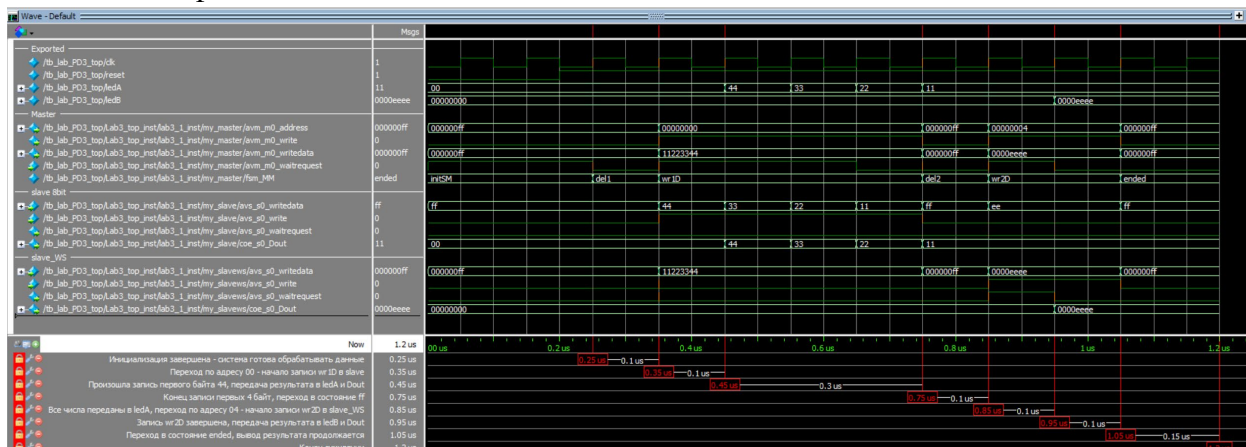


Рис. 34 – Моделирование проекта средствами ModelSim

master

Формирует записи по неким адресам. В начале это тот самый адрес ff, который является заполнителем, пока не поступит обычный адрес. Затем, он начинает формировать запись по 0-му адресу (проходит инициализация + задержка после начала моделирования) По 0-му адресу записываются значения, указанные в writedata (32'h11223344). 32 разряда
ВЫКИНУТО

my slave (8-битный)

32 разряда было выкинуто, но, поскольку этот slave – 8-битный, у него появляется сигнал wr1D и начинают формироваться данные (сначала младшие 44, потом 33, потом 22, наконец - 11). Во время этого у master возникает сигнал wr1D (wait request). Это означает, что в то время, пока байты 44, 33, 22, 11 последовательно передаются, мастер заторможен. Этот сигнал, Wr1D, сформировала система меж соединений (она увидела, что было подано 4 байта = 32 бита и надо притормозить master, пока производится запись этих байтов в 8-битный my slave).

my slaveWS (32-битный)

Далее осуществляется «пустой» цикл (со значениями ff). После этого master осуществляет запись по адресу = 4 (значение writedata = 32'h0000eeee). Здесь заметим, что 8-битный my_slave получает данные в виде `ee`, но у него нету никакого признака write, в отличие от 2-го (32-битного) my_slaveWS. Соответственно, только 2-ой slave получит значения, но он зафиксирует эти значения только в тот момент, когда master перестанет подавать сигнал waitrequest. В этот момент произойдёт фиксация этих значений и master перейдёт в состояние ended.

Таким образом, происходит последовательная передача в режиме little endian 32-битных данных с выставлением waitrequest'a. Происходит выставление waitrequest'a также и в режиме big endian, но уже по инициативе slave (32-битного).

4.2. Тестирование средствами Signal Tap II

Создание файла для отладки

Создадим файл db_lab_PD3_top.sv для отладки модуля lab_PD3_top:

```
lab_PD3 - db_lab_PD2_top.sv

1 module db_lab3_PD2_top (
2     (* altera_attribute = "-name IO_STANDARD \"3.3-V LVCMOS\"", chip_pin = "23" *)
3     input bit clk
4 );
5     bit reset;
6     bit [7:0] ledA;
7     bit [31:0] ledB;
8     SP_unit u0 (
9         .source      (reset),
10        .source_clk   (clk)
11    );
12    lab_PD3_top Lab3_top_inst (.*)
13 endmodule
14
```

Рис. 35 – Файл для отладки модуля верхнего уровня

Создадим модуль ISSPE, укажем файл db_lab_PD3_top.sv файлом верхнего уровня и убедимся в том, что схема, получаемая в результате компиляции, будет верной:

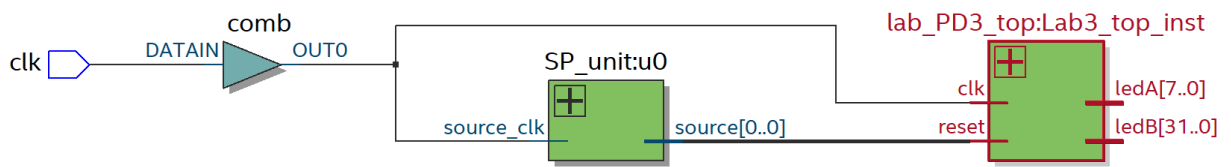


Рис. 36 – Схема проекта с добавлением SP_unit в RTL Viewer

Как видно из схемы SP_unit добавлен корректно.

Настройка Signal Tap II

Выберем сигналы для логического анализатора:

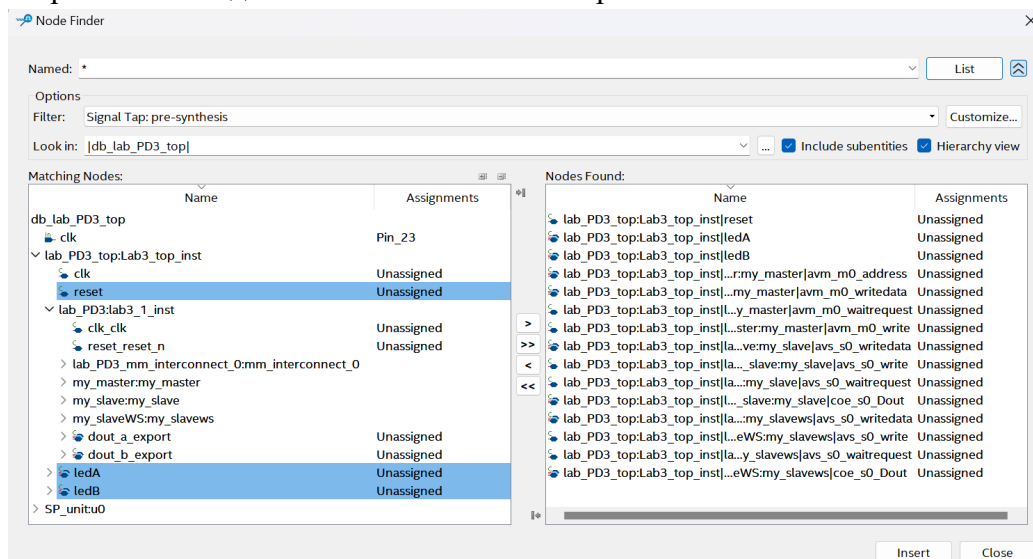


Рис. 37 – Сигналы логического анализатора

Добавим к сигналам конечный автомат:

State Machine - db_lab_PD3_top lab_PD3_top:Lab3_top_inst lab_PD3:lab3_1_inst my_master:my_master fsm_MM							
	Name	fsm_MM.ended	fsm_MM.wr2D	fsm_MM.del2	fsm_MM.wr1D	fsm_MM.del1	fsm_MM.initSM
1	fsm_MM.initSM	0	0	0	0	0	0
2	fsm_MM.del1	0	0	0	0	1	1
3	fsm_MM.wr1D	0	0	0	1	0	1
4	fsm_MM.del2	0	0	1	0	0	1
5	fsm_MM.wr2D	0	1	0	0	0	1
6	fsm_MM.ended	1	0	0	0	0	1

Рис. 38 – Конечный автомат

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

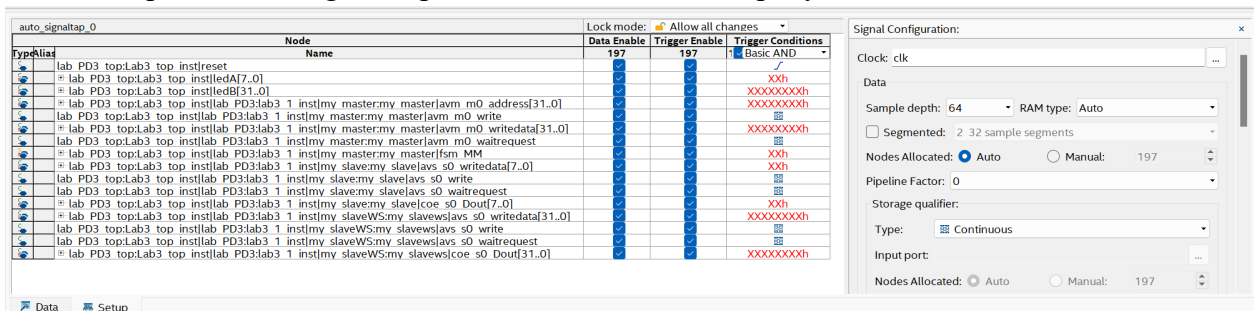


Рис. 39 – Настройка окна Signal Tap II

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				
	Fmax	Restricted Fmax	Clock Name	Note
1	60.81 MHz	60.81 MHz	altera_reserved_tck	

Рис. 40 – Временные характеристики устройства

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

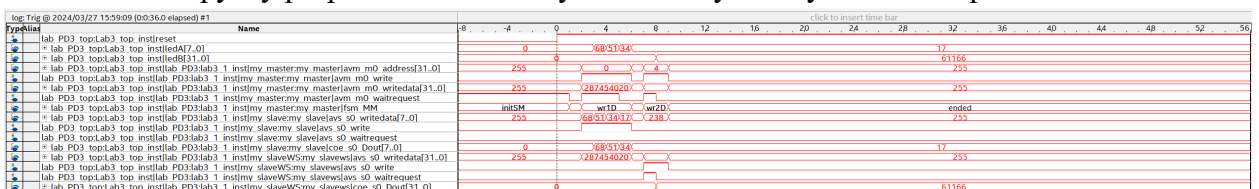


Рис. 41 – Результат SignalTap II

Полученная временная диаграмма совпадает с той, что была получена в ходе тестирования проекта средствами ModelSim (Рис. 34). Данные поступают и передаются на приёмник корректно.

5. Дополнительное задание

5.1. Изменение значений по варианту

Изменение адресов slave

В соответствии с вариантом в списке группы (6 вариант) установим значения адресов для модулей $my_slave = 24_{10} = 18_{16}$ и $my_slaveWS = 48_{10} = 30_{16}$ соответственно.

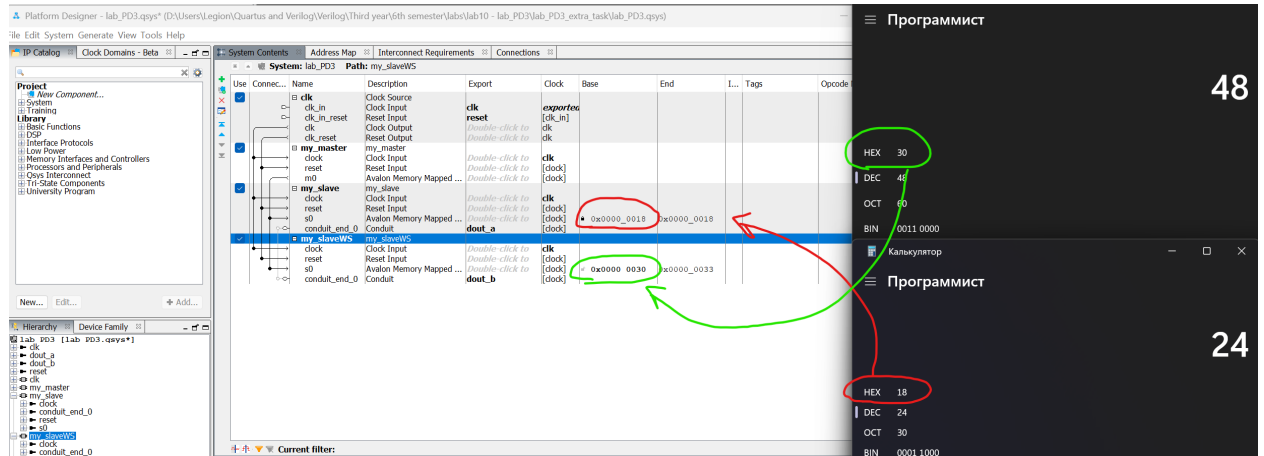


Рис. 42 – Изменение адресов для slave'ов

Окно Address Map будет выглядеть следующим образом:

System Contents		Address Map		Interconnect Requirements		Connections	
System: lab_PD3		Path: clk					
		my_master.m0					
my_slave.s0		0x0000_0018	-	0x0000_0018			
my_slaveWS.s0		0x0000_0030	-	0x0000_0033			

Рис. 43 – Изменённое окно Address Map

Изменение компонента master

Так как модуль master работает вместе со slave'ами, необходимо также поменять адреса внутри самого компонента master, установив следующие значения:

- $address_1 = 24_{10}$
- $data_1 = 287454020$ (не изменяется)
- $address_2 = 48_{10}$
- $data_2 = 6$ (номер варианта)

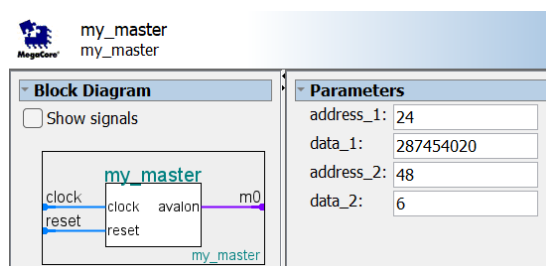


Рис. 44 – Изменённый модуль master

5.2. Тестирование средствами ModelSim

Симуляция средствами ModelSim

Выполним компиляцию проекта средствами ModelSim. Для этого запустим waveLab2.do файл:

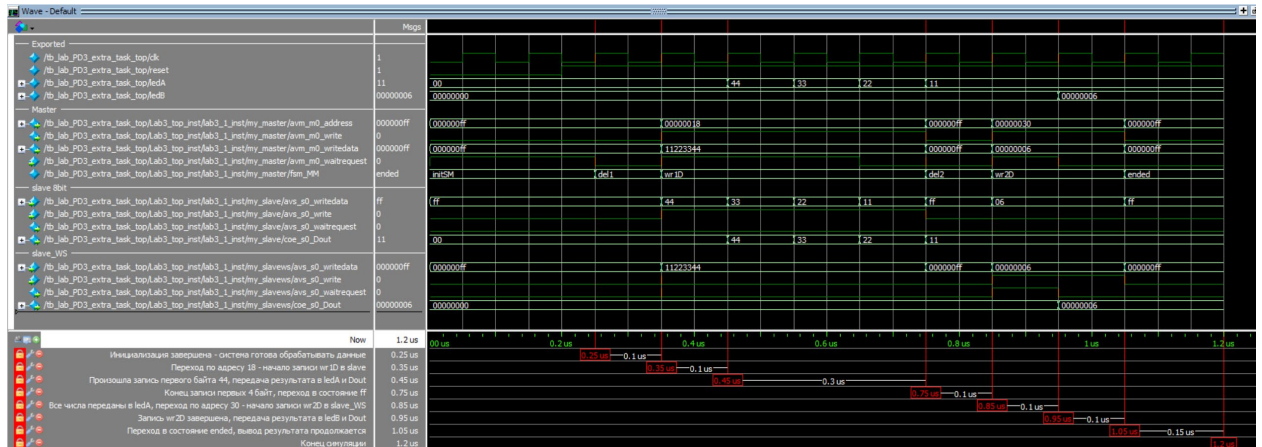


Рис. 45 – Моделирование проекта средствами ModelSim

Заметим, что временная диаграмма корректно обрабатывает изменённые адреса и значения, запись wr1D теперь происходит по адресу $24_{10} = 18_{16}$ (происходит запись данных $data_1 = 11223344$, это значение мы оставляли таким же, как и в первой симуляции), а запись wr2D по адресу $48_{10} = 30_{16}$ (осуществляется запись данных $data_2 = 6$, это значение согласно номеру варианта). Для удобства добавлены курсоры с пометками о том, что происходит на каждом из этапов, так, например становится понятно, что запись одного байта длится 0.1 мкс (или 100 ns). Отсюда следует, что цикл записи $my_slave = 0.4$ мкс (или 400 ns), $my_slaveWS = 0.2$ мкс (или 200 ns). Также, видим, что задержка обработки для некоторых сигналов составляет 0.1 мкс (или 100 ns).

5.3. Тестирование средствами Signal Tap II

Тестирование на плате средствами Signal Tap II

Настройки Signal Tap II остаются такими же, единственное, что изменяется, некоторые адреса сигналов, так как для дополнительного задания был создан отдельный проект.

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	60.71 MHz	60.71 MHz	altera_reserved_tck	

Рис. 46 – Временные характеристики устройства

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

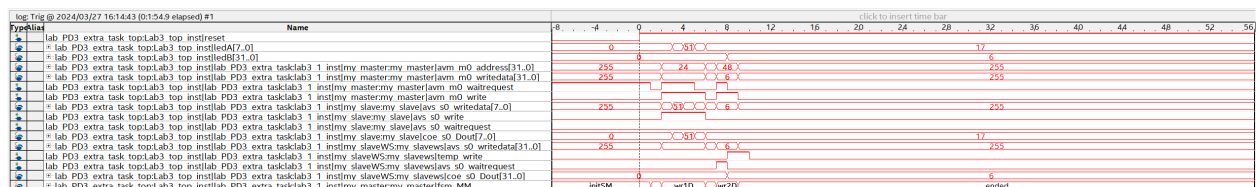


Рис. 47 – Результат SignalTap II

Полученная временная диаграмма совпадает с той, что была получена в ходе тестирования проекта средствами ModelSim (Рис. 34). Данные поступают и передаются на приёмник корректно.

6. Вывод

В ходе лабораторной работы успешно разработан и скомпилирован проект в среде ModelSim, включающий в себя master-модуль и два slave-модуля: my_slave и my_slaveWS. Анализ временной диаграммы подтвердил корректность передачи данных между модулями.

В `my_slave`, являющемся 8-битным устройством, подается 32 бита данных, что требует организации очереди для их записи. Именно поэтому возникает необходимость в использовании сигнала wr1D, чтобы обработать эти данные порциями по 8 бит.

С другой стороны, второй slave (my_slaveWS) является 32-битным устройством, что позволяет обрабатывать 32-битные данные сразу, без необходимости в организации очереди. Поэтому для него используется сигнал wr2D, и данные обрабатываются непосредственно.

Внесенные изменения в адреса и данные были успешно учтены в проекте, и временная диаграмма верно отображает процесс обмена данными между master- и slave-модулями, учитывая их особенности и требования к обработке данных.