

Приложение Platform Designer

Приложение Platform Designer

Часть 3

Интерфейсы и компоненты шины Avalon-MM

План

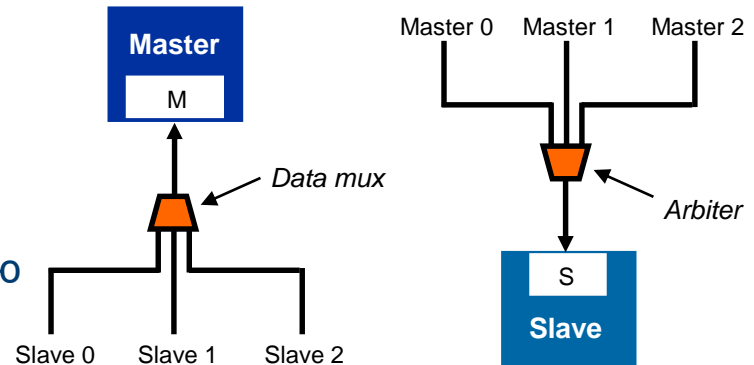
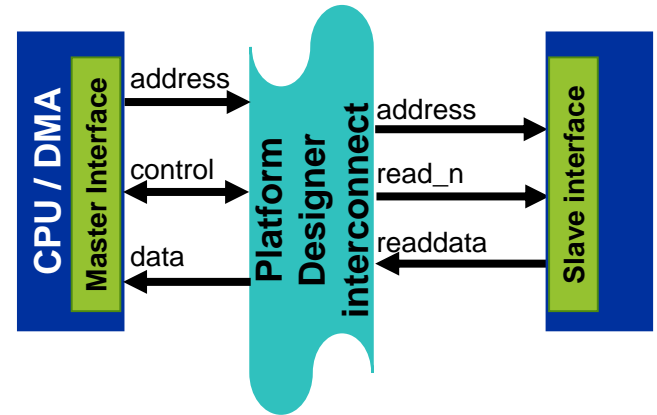
- Интерфейсы шины Avalon-MM
- Типы обменов на шине Avalon-MM
- Адресация на шине Avalon-MM
- Компоненты шины Avalon-MM
- Лабораторная 3

Интерфейсы шины Avalon-MM

- Шина Avalon-MM – адресуемая (memory-mapped) шина, обеспечивающая обмен между **Ведущим (Master)** и **Ведомым (Slave)** в режиме Чтения/Записи.
- На шине определены интерфейсы
 - Интерфейс **Master** – интерфейс Ведущего
 - Ведущий (Master) инициирует запросы чтения/записи на шине по адресу из адресного пространства
 - Интерфейс **Slave** – интерфейс Ведомого
 - Ведомый (Slave) привязан к адресу в адресном пространстве Ведущего
 - Принимает адресованные ему запросы и отвечает на них
- Все обмены по шине синхронные
 - Все интерфейсы должны быть привязаны к интерфейсу Clock
- Опционально, но рекомендовано, что бы ко все интерфейсам был привязан интерфейс Reset (если не привязан – PD формирует предупреждение)

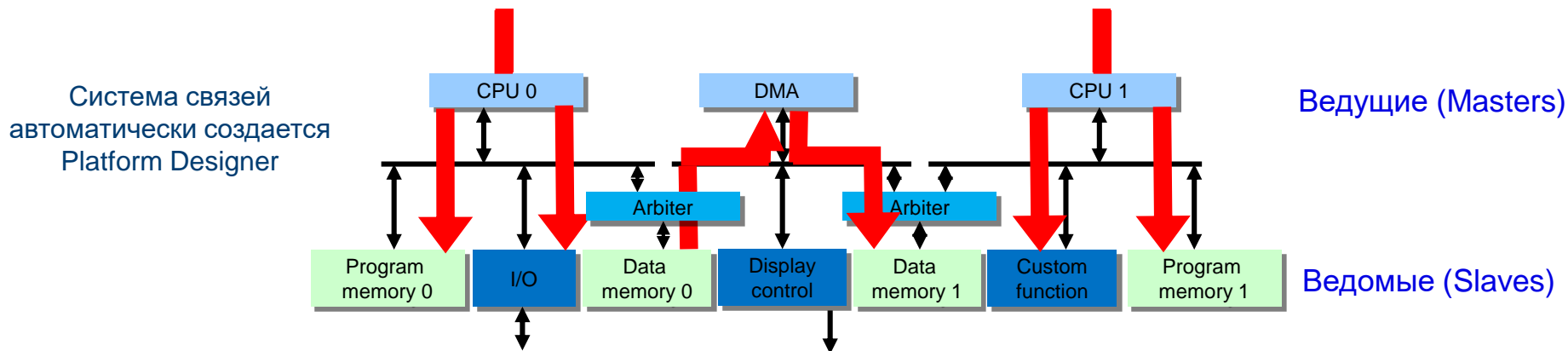
Принципы реализации шины Avalon-MM в PD

- Адресация (Memory addressing)
 - PD создает адресный декодер, обеспечивающий Ведущему возможность доступа к регистрам Ведомого
- Мультиплексирование данных (Data multiplexing)
 - PD создает мультиплексор на входе Ведущего, обеспечивающий возможность доступа Ведущего к нескольким Ведомым
- Арбитраж
 - PD создает схему арбитража (арбитра) на входе Ведомого, который находится в адресном пространстве нескольких Ведущих, управляющую доступом к Ведомому



Система с Multi-Mastering

- Система связей, реализуемая на ресурсах FPGA:
 - Обеспечивает арбитраж на стороне Ведомого (slave-side arbitration)
 - Возможна реализация нескольких одновременных обменов по шине если они реализуются к разным Ведомым (реализуется несколько доменов)
 - Trade-off: увеличиваются использованные аппаратные ресурсы
 - Пример: лабораторная работа 1



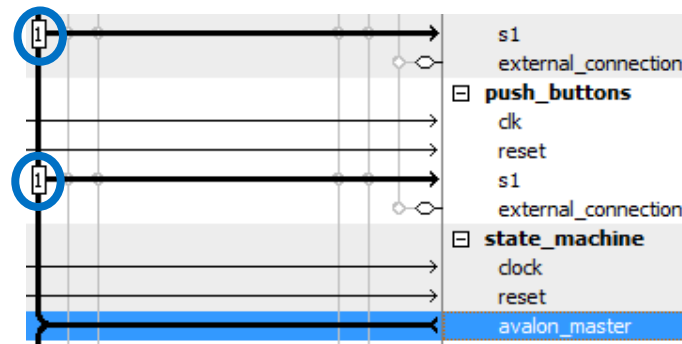
Базовый способ арбитража: Round-Robin

По умолчанию система связей реализует арбитраж round-robin с одинаковыми долями

- Каждое соединение master/slave получает фиксированную «долю» передач для обмена
 - По умолчанию для всех установлена равная доля = 1
 - Передача Burst считается как одна доля
- При возникновении конфликта приоритет имеет Ведущий с большей долей
- Если использованы не все доли, то они теряются и управление передается следующему Ведущему.

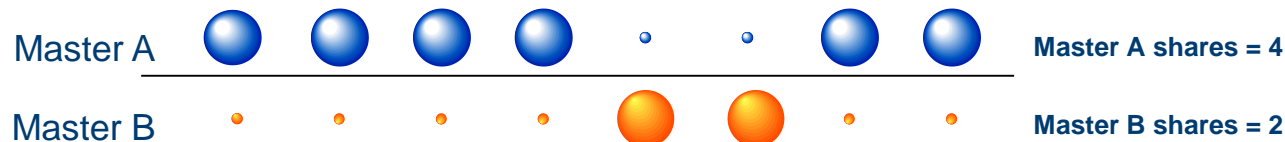
Назначение долей для арбитража

- Выберите master/slave соединение в закладке Hierarchy tab и назначьте долю в закладке Parameters tab
- Right-click в столбце Connections и выберите Show Arbitration Shares



- Пример: masters A/B подключены к одному slave

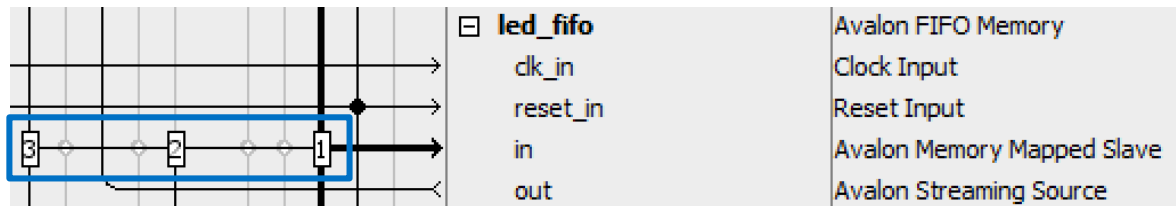
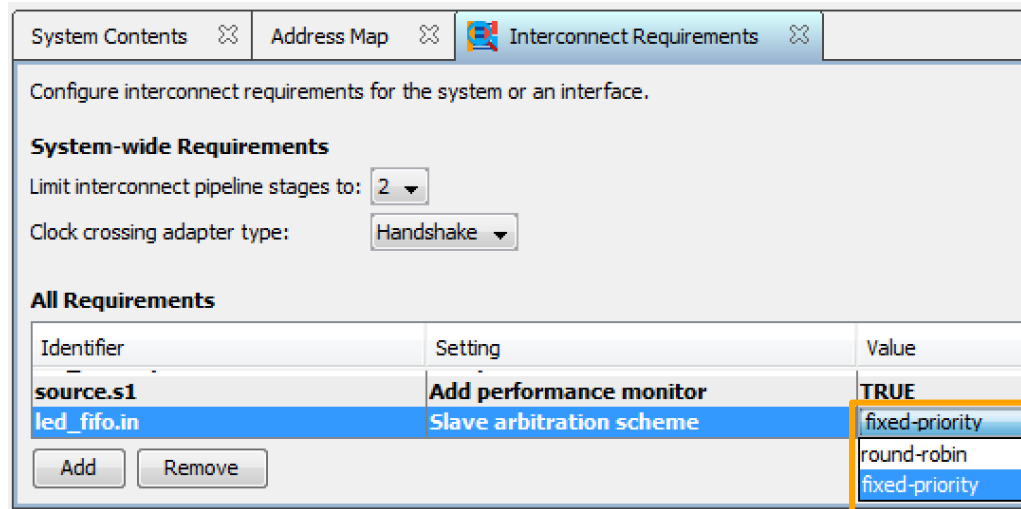
Arbiter



Арбитраж с фиксированным приоритетом

Назначение фиксированных приоритетов для доступа к Ведомому от Ведущих

- Доступ получает Ведущий с более высоким приоритетом – другие ожидают
- Выбор схемы арбитража на закладке Interconnect Requirements
- В столбце Connections установите значения приоритета для каждого мастера
 - Все значения должны быть уникальными



Сигналы интерфейса Avalon-MM Master

Имя	Разрядность	Направление	Обязательно использовать	Описание
address	1-64	Output	Y	Byte address corresponding to slave for transfer request; must align with data width
waitrequest waitrequest_n	1	Input	Y	Forces master to stall transfer until deasserted; other Avalon-MM interface signals must be held constant
read read_n	1	Output	N	Indicates master issuing read request
readdata	8, 16, 32, 64, 128, 256, 512, 1024	Input	N	Data returned from read request
write write_n	1	Output	N	Indicates master issuing write request
writedata	8, 16, 32, 64, 128, 256, 512, 1024	Output	N	Data to be sent for write request
byteenable byteenable_n	2, 4, 8, 16, 32, 64, 128	Output	N	Specifies valid byte lane(s) for readdata or writedata (width = data width / 8)
lock	1	Output	N	Once master is granted access to shared slave, locks arbiter to master until deasserted
response	2	Input	N	Indicates successful transfer or not, as well as whether access is to undefined address location

Сигналы интерфейса Avalon-MM Slave

Имя	Разрядность	Направление	Обязательно использовать	Описание
address	1-64	Input	N	Word address of slave for transfer request (<i>discussed later</i>)
waitrequest waitrequest_n	1	Output	N	Allows slave to stall transfer until deasserted; other Avalon®-MM interface signals must be held constant
read read_n	1	Input	N	Indicates slave should respond to read request
readdata	8, 16, 32, 64, 128, 256, 512, 1024	Output	N	Data provided to Platform Designer interconnect in response to read request
write write_n	1	Input	N	Indicates slave should respond to write request
writedata	8, 16, 32, 64, 128, 256, 512, 1024	Input	N	Data from the Platform Designer interconnect for a write request
byteenable byteenable_n	2, 4, 8, 16, 32, 64, 128	Input	N	Specifies valid byte lane for readdata or writedata (width = data width / 8)
begintransfer begintransfer_n	1	Input	N	Asserts at the beginning (first cycle) of any transfer
response	2	Output	N	Indicates successful transfer or not, as well as whether access is to undefined address location; interconnect provides OK if no slave response

Минимально необходимый набор сигналов

- Интерфейс **Master**

- Обязательно наличие **address** и **waitrequest**
- Для запроса **read** необходимо наличие **read** и **readdata**
- Для запроса **write** необходимо наличие **write** и **writedata**

- Интерфейс **Slave**

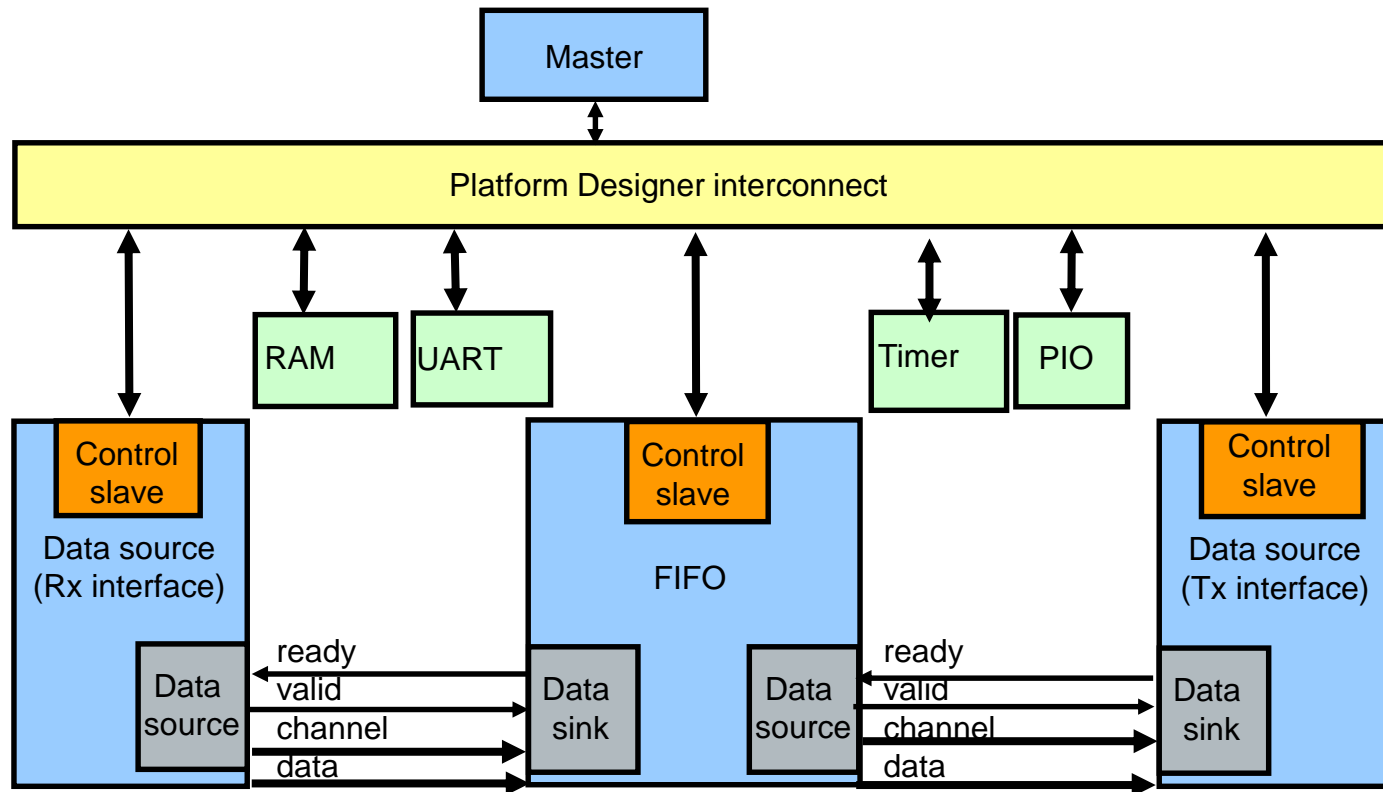
- Не требуется **address**;
 - Декодирование базового адреса, заданного в PD обеспечивает, адресный декодер, реализуемый PD в системе связей компонентов
- Не требуется **waitrequest**;
- Для ответа на запрос **read** необходимо наличие **readdata**
- Для ответа на запрос **write** необходимо наличие **write** и **writedata**

Другие интерфейсы Avalon

- Avalon-C (conduit)
 - Группа произвольных сигналов, которые предназначены для экспорта как выводы системы
- Avalon-TC (tristate conduit)
 - Point-to-point интерфейс для on-chip контроллеров, управляющих off-chip компонентами по шине с Z состоянием (tri-state buses)
 - Например: Address, data, control выводы могут быть подключены к нескольким tri-state устройствам для экономии выводов
- Avalon interrupt
 - Interrupt sender: отправляет одноразрядный irq к receiver
 - Interrupt receiver: получает irq (разрядность входа до 32 бит) от sender

Использование Avalon-MM и Avalon-ST

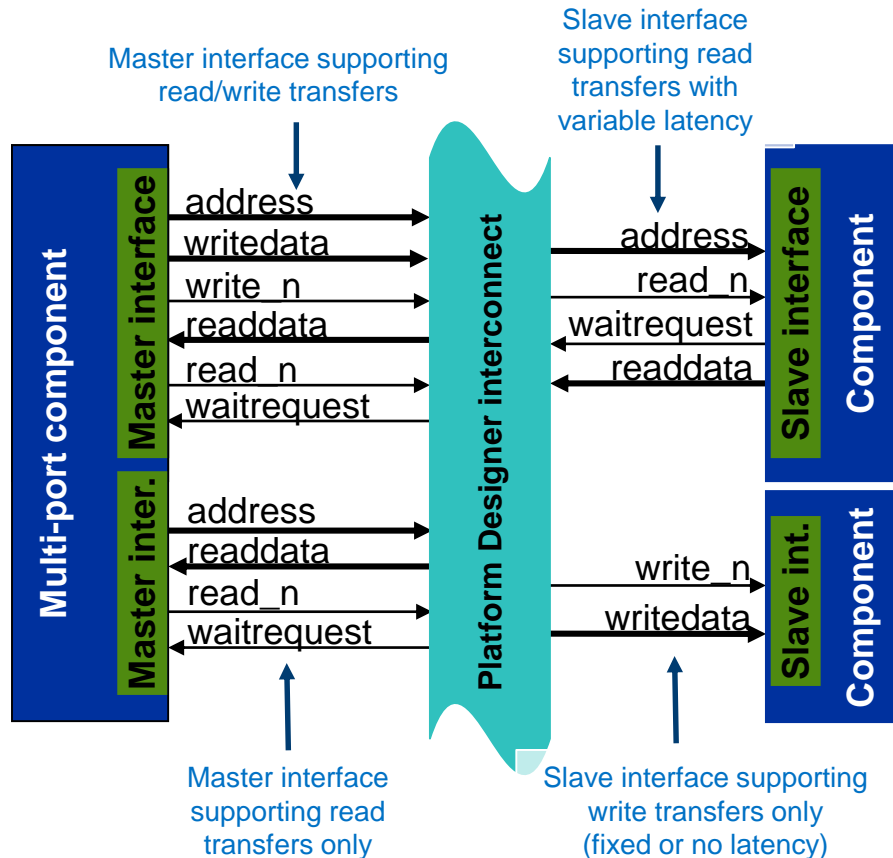
Интерфейсы Avalon-MM и Avalon-ST дополняют друг друга



План

- Интерфейсы шины Avalon-MM
- **Типы обменов на шине Avalon-MM**
- Адресация на шине Avalon-MM
- Компоненты шины Avalon-MM
- Лабораторная 3

Пример интерфейсов на шине Avalon-MM

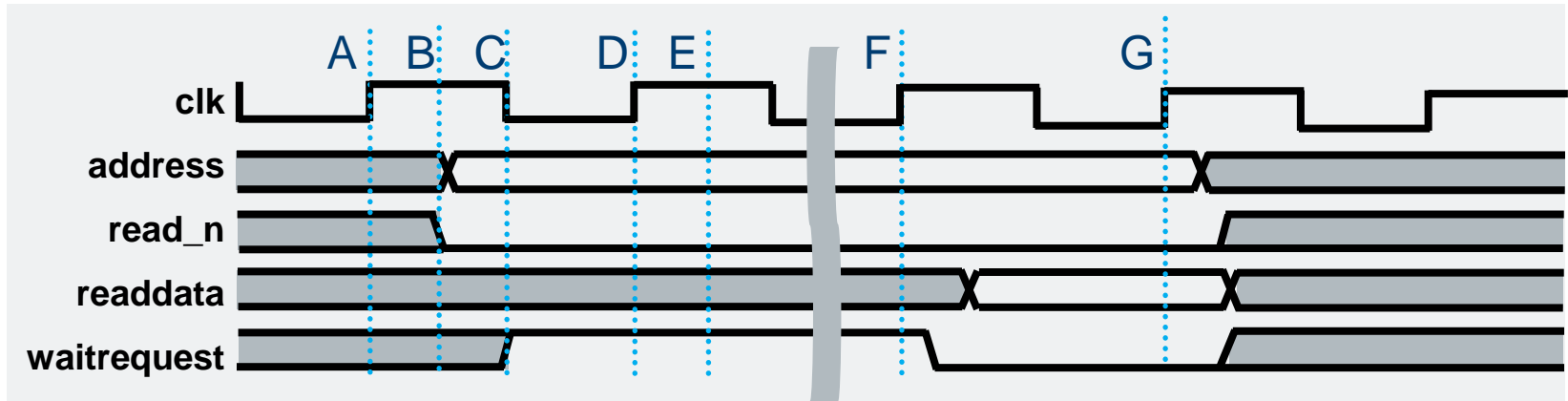
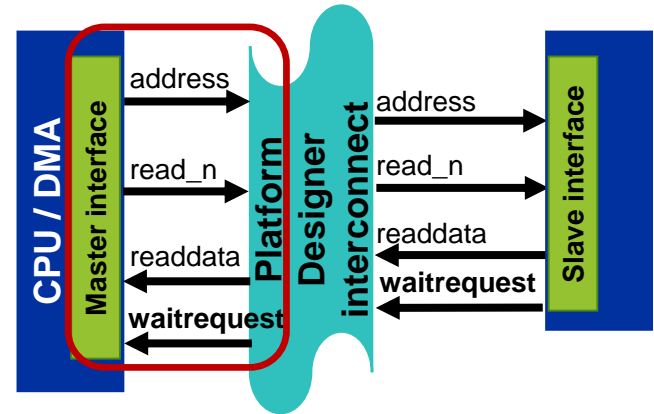


Типы обменов на шине Avalon-MM

- Основные типы обменов на шине
 - Master read/write
 - Slave read/write
- Обмен с контролем задержки
 - Фиксированная задержка - Fixed latency (wait states)
 - Переменная задержка - Variable latency (component-controlled wait states)
- Ускоренный обмен
 - Pipelined
 - Burst

Обмен Master Read с переменной задержкой

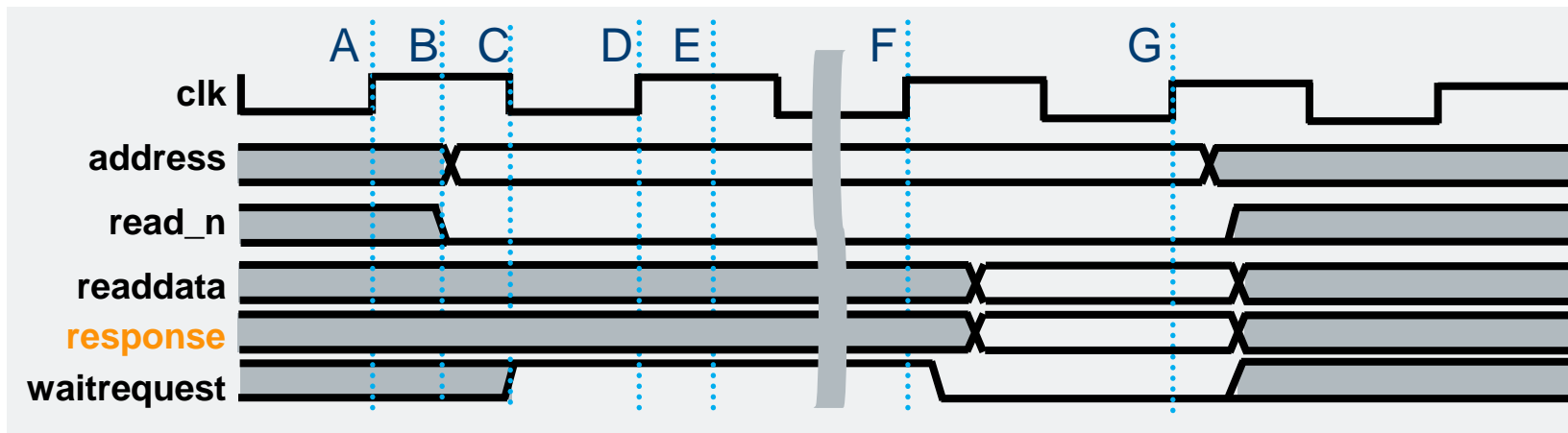
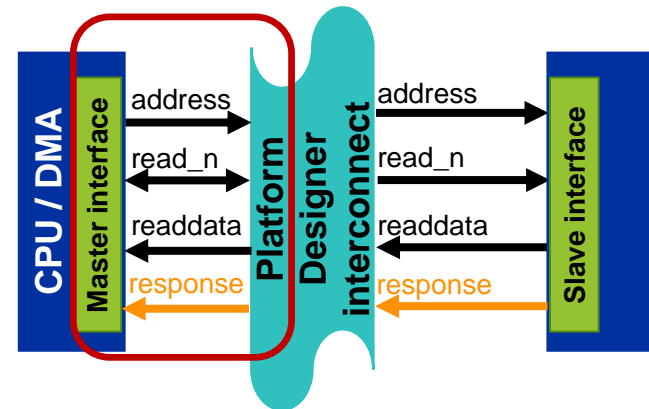
- Устанавливаются **address** & **read_n**
- Ожидание **waitrequest** = '0'
- Master фиксирует **readdata** на следующем фронте тактового сигнала
- Окончание обмена



Обмен Master Read с переменной задержкой

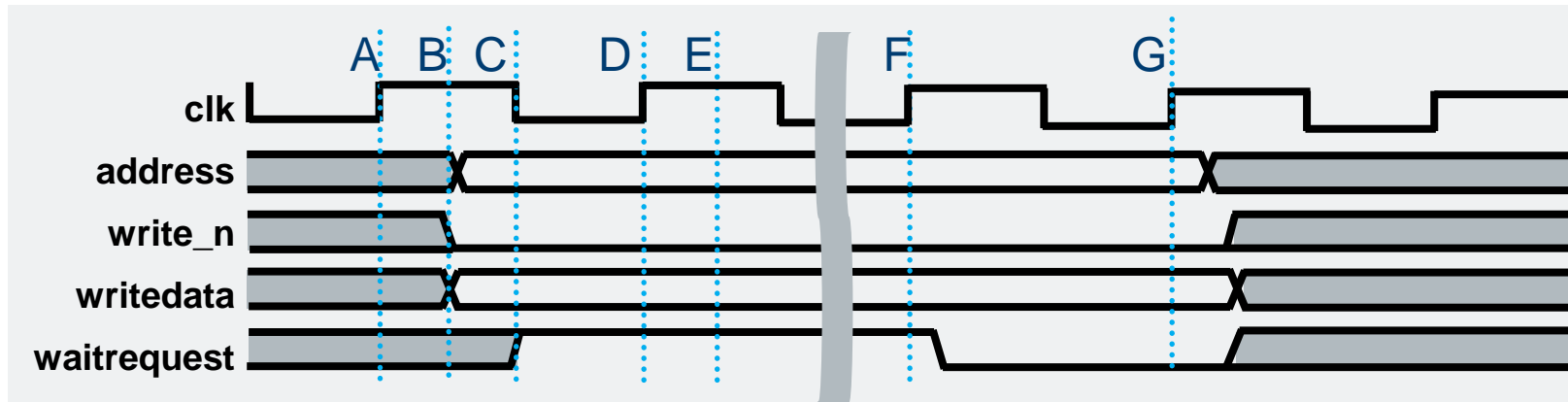
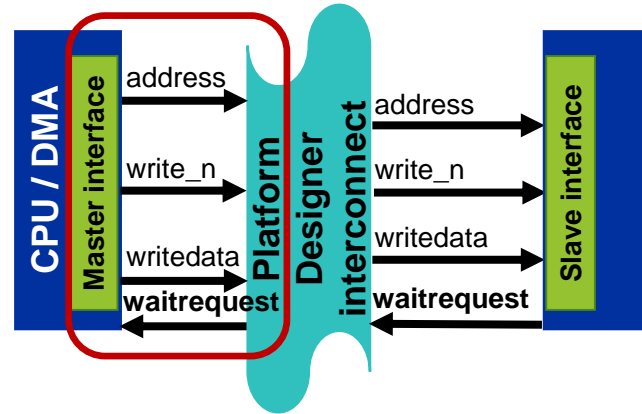
▪ **response** передается вместе с readdata

- 2'b00: OKAY передача OK
- 2'b10: SLAVEERROR ошибка
- 2'b11: DECODEERROR доступ к адресу вне карты памяти



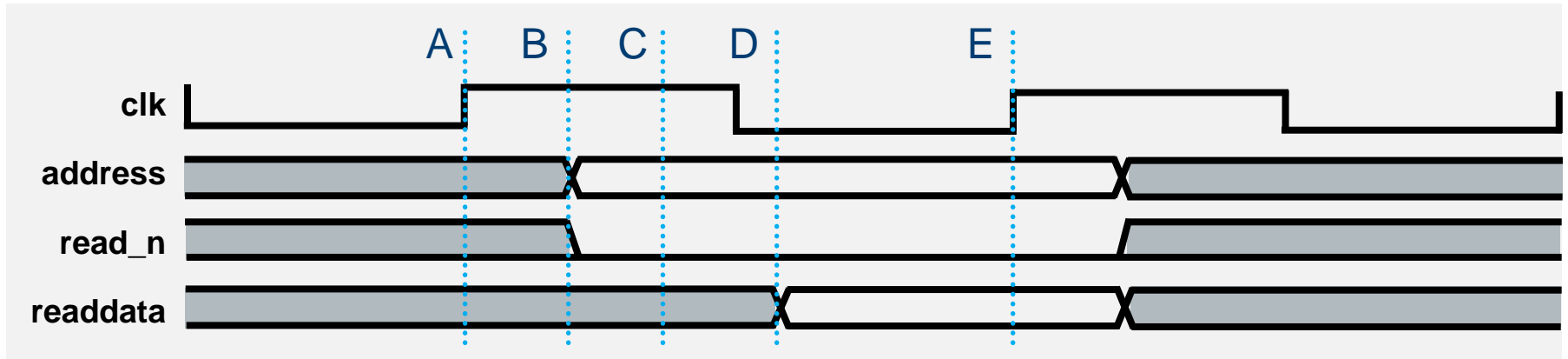
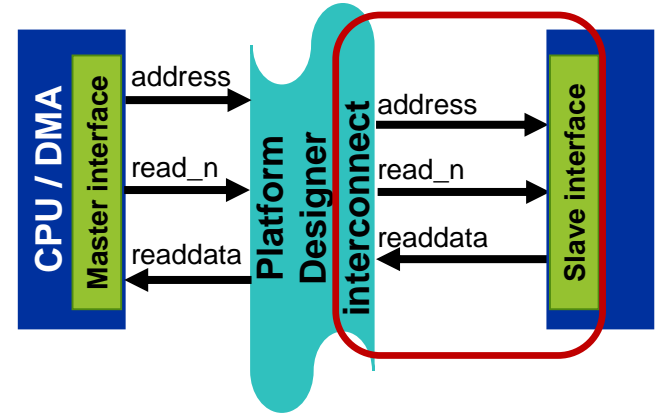
Обмен Master Write с переменной задержкой

- Устанавливаются address, write_n, & writedata
- Ожидание waitrequest = '0'
- Обмен заканчивается на следующем фронте тактового сигнала



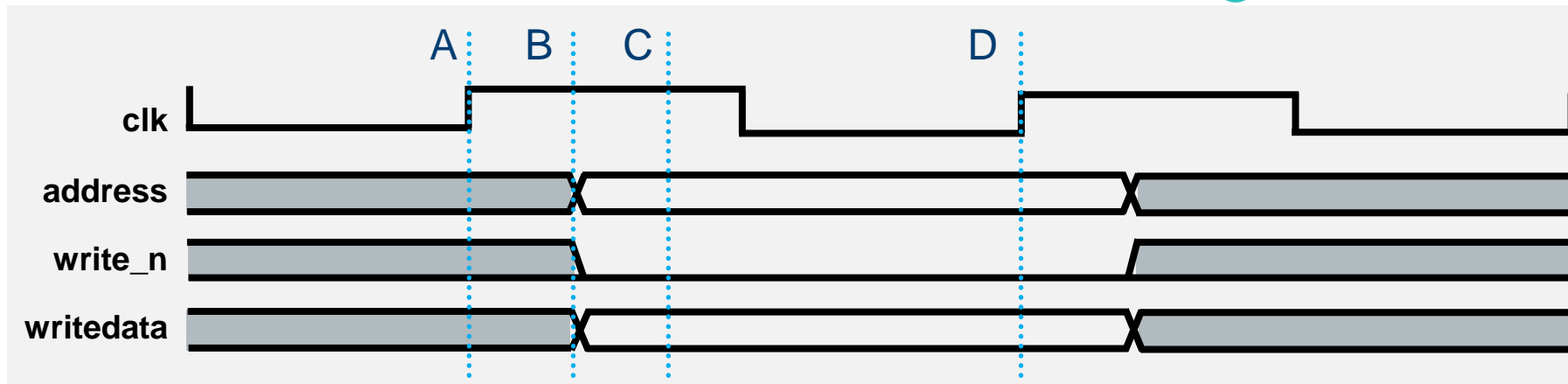
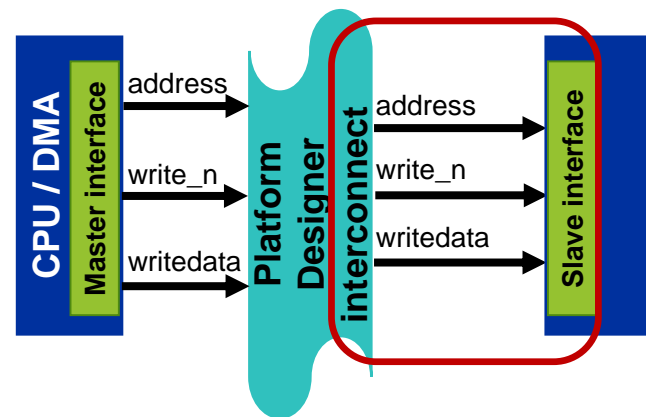
Обмен Slave Read без задержки

- Устанавливаются `address` & `read_n`
- Ведомый Slave передает `readdata` к следующему фронту тактового сигнала
 - 0 wait states
- Обмен заканчивается



Обмен Slave Write без задержки

- Устанавливаются address, write_n, & writedata
- Ведомый (Slave) фиксирует writedata на следующем фронте тактового сигнала
 - 0 wait states
- Обмен заканчивается



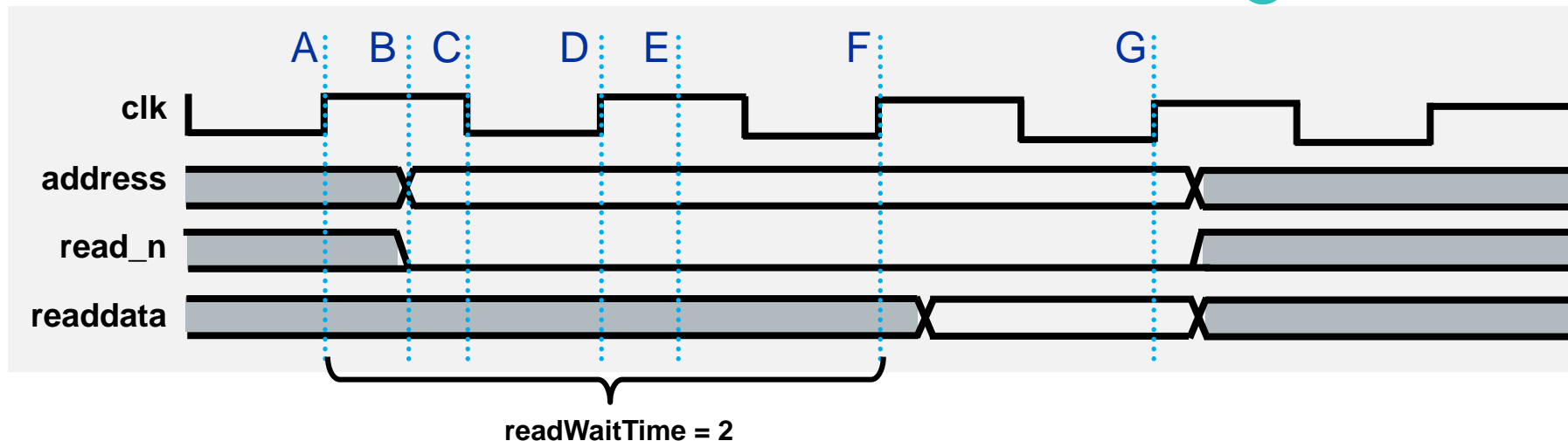
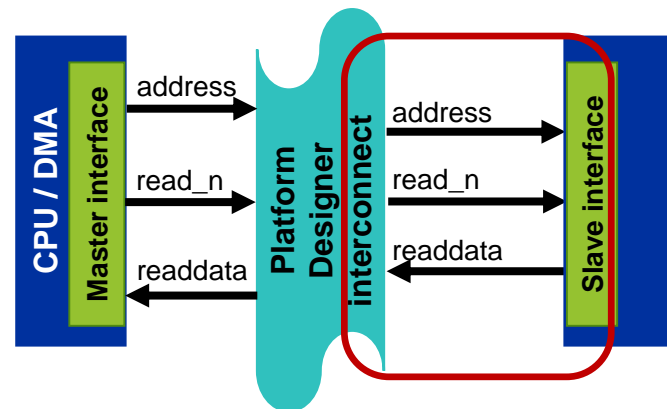
Параметры обмена Avalon-MM Slave

PD позволяет настроить параметры обмена для некоторых библиотечных (off-the-shelf) IP и для пользовательских компонентов

- readWaitTime / writeWaitTime
 - Допустимые значения: 0 - 1000
 - Число тактов (нс) до
 - Фиксации Ведомым записываемых данных
 - Формирования Ведомым читаемых данных
 - Используется для обмена с фиксированной задержкой (Ведомый не имеет сигнала waitrequest)
- timingUnits
 - Допустимые значения : cycles or ns
 - Единица измерения для readWaitTime и writeWaitTime
- Параметры описаны в Intel® Quartus® Prime software handbook

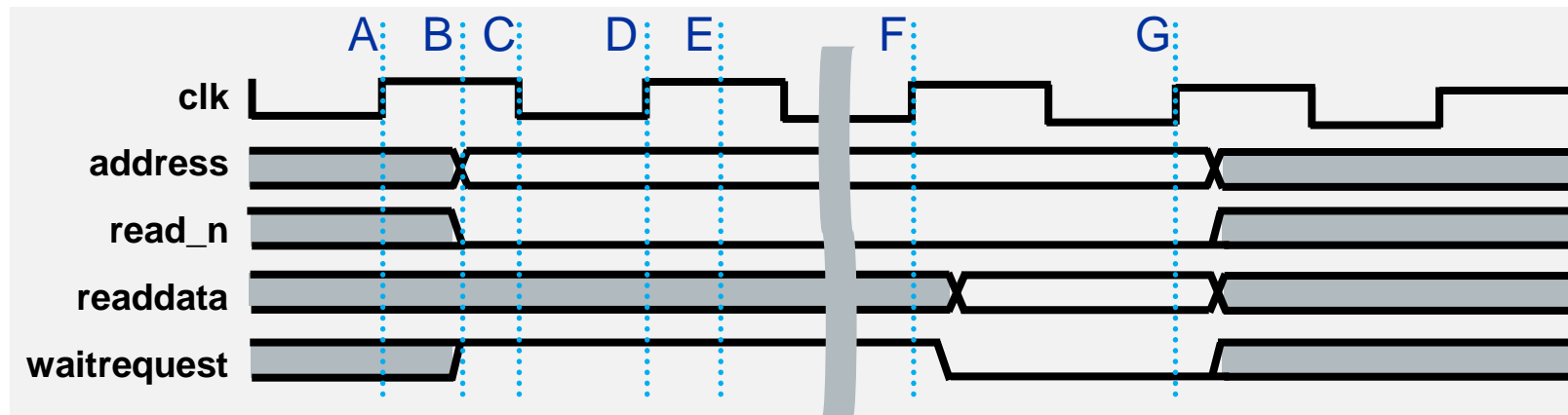
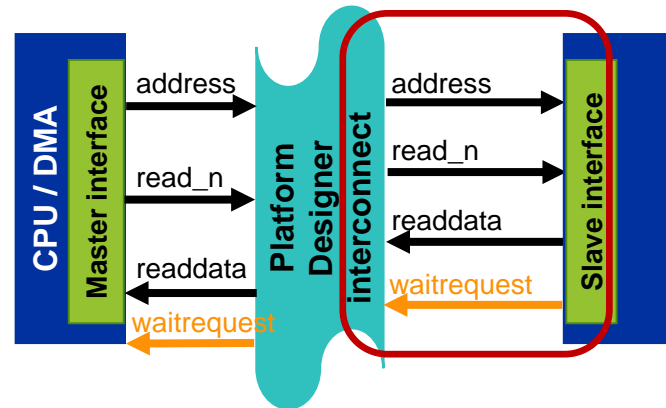
Обмен Slave Read с фиксированной задержкой

- Устанавливаются address & read_n
- После заданного параметром readWaitTime числа тактов Ведомый возвращает readdata к следующему фронту тактового сигнала
- Обмен заканчивается

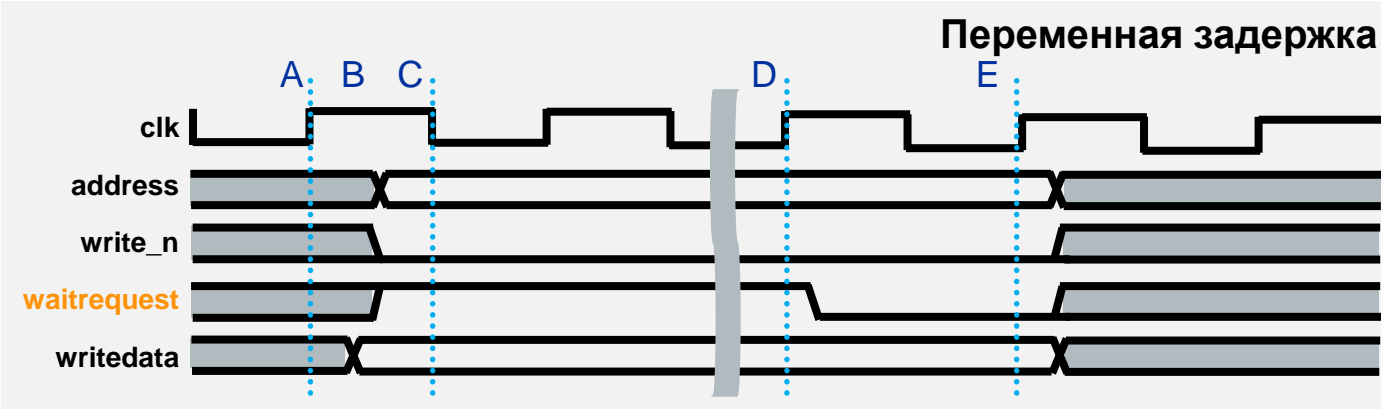
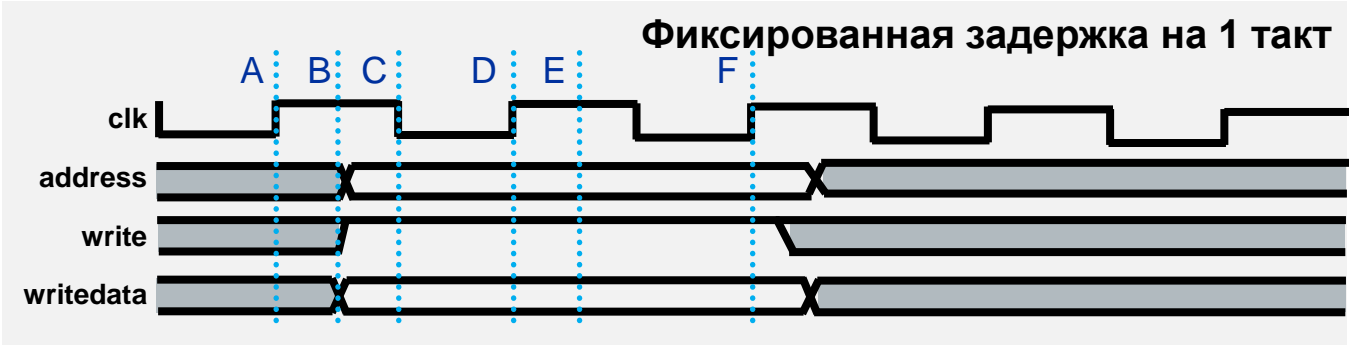


Обмен Slave Read с переменной задержкой

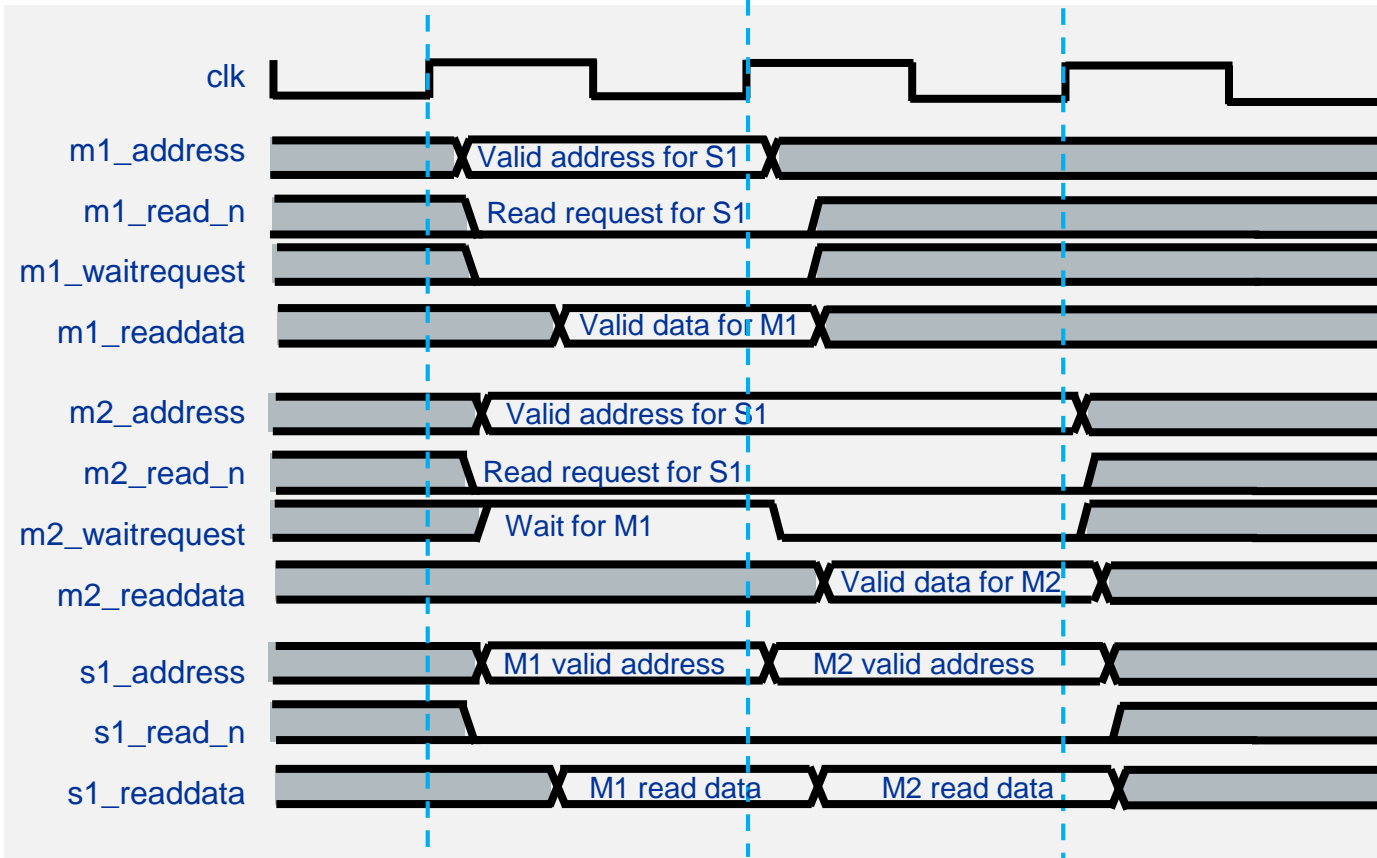
- Устанавливаются `address` & `read_n`
- Ведомый устанавливает `waitrequest` для приостановки обмена
- Ведущий фиксирует `readdata` по следующему ближайшему фронту после сброса `waitrequest`
- Обмен заканчивается



Обмен Slave Write при разных типах задержки



Обмен нескольких Masters с общим Slave



Ускоренный обмен – сигналы Master

Ускоренный обмен – дополнительные сигналы Master интерфейса

Имя	Разрядность	Направление	Обязательно использовать	Описание
Bursting transfers				
burstcount	1-11	Output	N	Indicates number of transfers in burst (# transfers = $2^{(\text{width} - 1)}$); if bursts are reads, must include readdatavalid signal below
Pipelined transfers				
readdatavalid readdatavalid_n	1	Input	N	Indicates valid data from prior read transfer request is available on readdata input; only used with variable pipeline latency
writeresponsevalid	1	Input	N	Used along with response to indicate a write response one cycle (by default) after a write; increase with minimumResponseLatency property

Ускоренный обмен – сигналы Slave

Ускоренный обмен – дополнительные сигналы Slave интерфейса

Имя	Разрядность	Направление	Обязательно использовать	Описание
Bursting transfers				
burstcount	1-11	Input	N	Indicates number of transfers in burst (# transfers = $2^{(\text{width} - 1)}$)
beginbursttransfer	1	Input	N	Asserted during first cycle of burst to indicate burst is starting; <i>not recommended to use this signal since slave should be able to calculate start of burst by counting transactions; available only to support legacy memory controllers</i>
Pipelined transfers				
readdatavalid readdatavalid_n	1	Output	N	Indicates valid data from prior read transfer request is available on readdata input; only used with variable pipeline latency
writeresponsevalid	1	Input	N	Used along with response to indicate a write response one cycle (by default) after a write; increase with minimumResponseLatency property

Ускоренный обмен: Pipelined

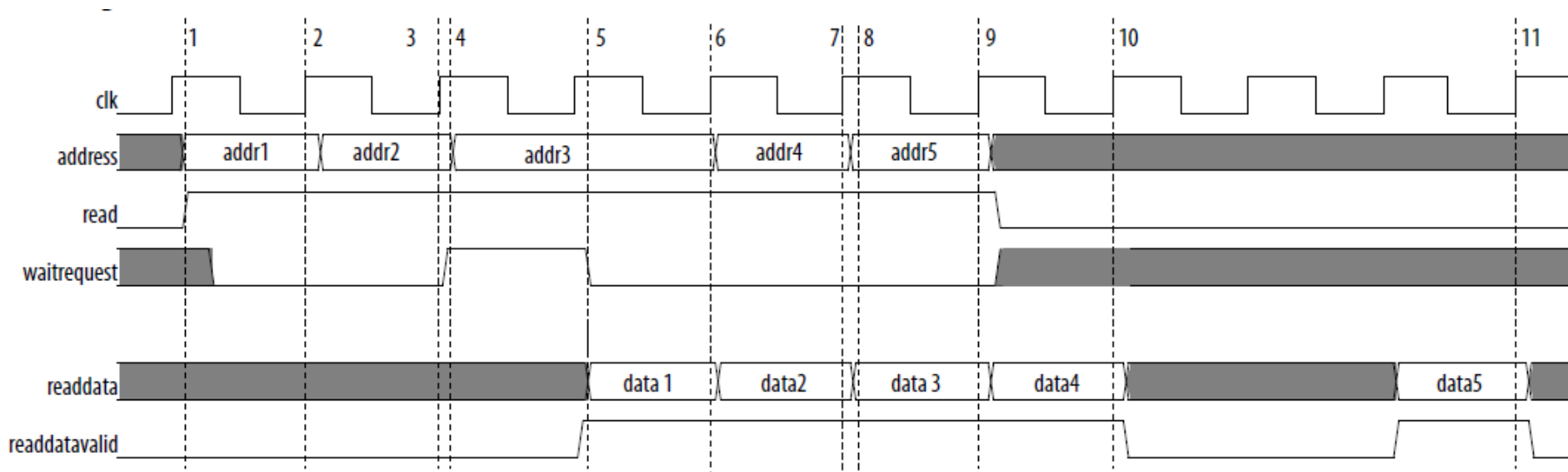
- Используется Ведущим и Ведомым поддерживающими несколько одновременных запросов read
 - Позволяет инициировать новый запрос read до окончания предыдущего запроса read
 - Запросы Writes не поддерживаются
- Разделяет обмен read на фазы address и data
 - Фаза Data phase наступает после фиксированной или переменной задержки относительно фазы address phase - это pipeline latency

Ускоренный обмен: Pipelined (задержки)

- Фиксированная задержка Fixed latency
 - Параметр `readWaitTime` - фиксированная задержка от фазы `address` фазы `data`
 - Может задаваться дополнительная задержка во время фазы `address` с помощью сигнала `waitrequest`
- Переменная задержка Variable latency
 - Контролируется сигналом `readdatavalid`
 - Сигнал `waitrequest` позволяет Ведомому контролировать поддерживаемое Ведомым число последовательных запросов

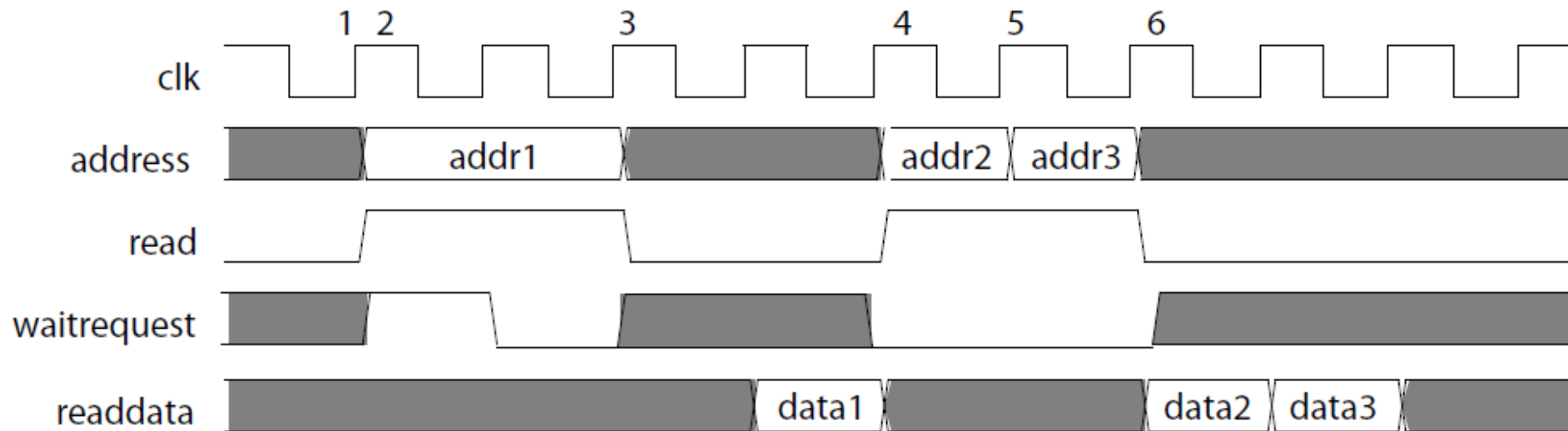
Пример Ускоренный обмен: Pipelined Read

Ускоренный обмен Pipelined Read с переменной задержкой



Пример Ускоренный обмен: Pipelined Read

Ускоренный обмен Pipelined Read с фиксированной задержкой=2



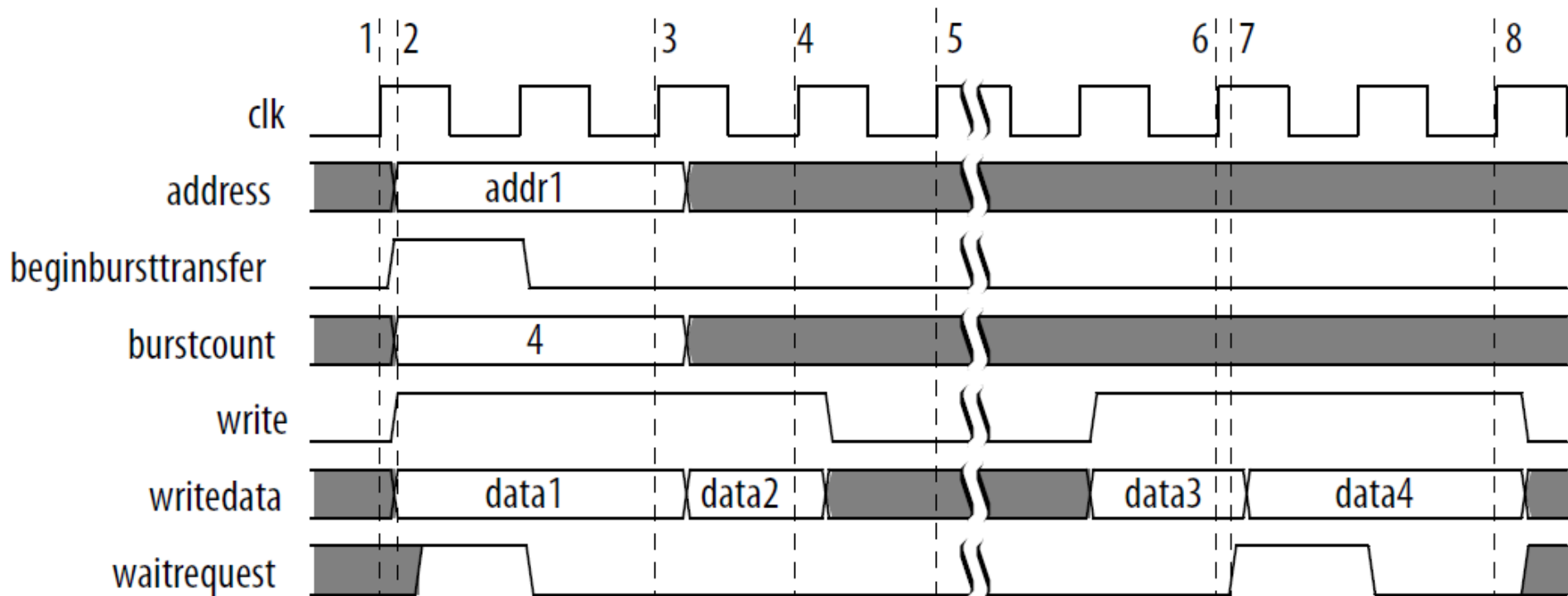
Ускоренный обмен: Bursting

Обеспечивает Ведущему непрерывный обмен с Ведомым для заданного числа передач

- Ведущий передает начальный адрес (starting address) и число передач (burstcount)
 - Ведомый должен обеспечить reading/writing данных с последовательной адресацией
- Арбитр автоматически предоставляет доступ Ведущему до завершения заданного числа передач
- Весь обмен (все заданное число передач) – одна доля арбитража
- Ведомый должен управлять задержкой используя сигнал waitrequest
- Для «read bursts» используется сигнал readdatavalid

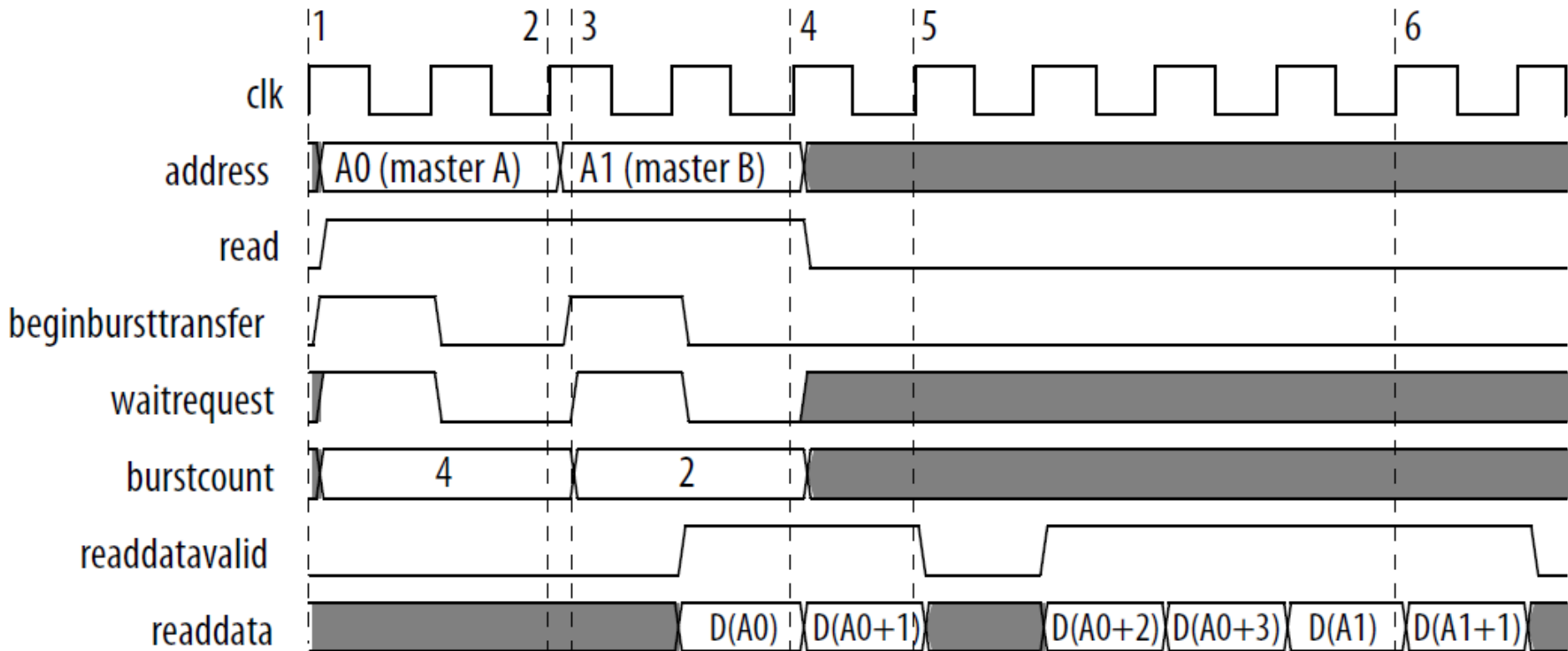
Пример Write Burst

Ускоренный обмен с параметром **constantBurstBehavior=false** для Ведущего и В ведомого



Пример Read Burst

Ускоренный обмен двух Ведущих с одним Ведомым



План

- Интерфейсы шины Avalon-MM
- Типы обменов на шине Avalon-MM
- Адресация на шине Avalon-MM
- Компоненты шины Avalon-MM
- Упражнение 3

Базовая адресация

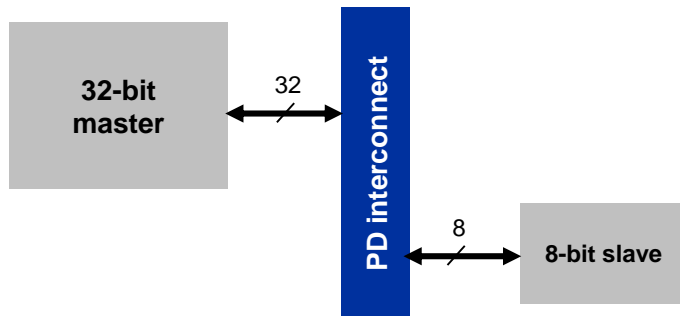
- Ведомый (Slaves) использует адресацию словами (слово имеет разрядность data)\
- Ведущий (Masters) использует байтовую адресацию (адресует каждый байт)
 - Счетчик адреса в 32-bit Ведущем должен увеличиваться на 4 для доступа к следующему слову 32-bit Ведомого



- PD позволяет установить Ведущему адресацию словами, а Ведомому адресацию байтами

Динамическое выравнивание адресов

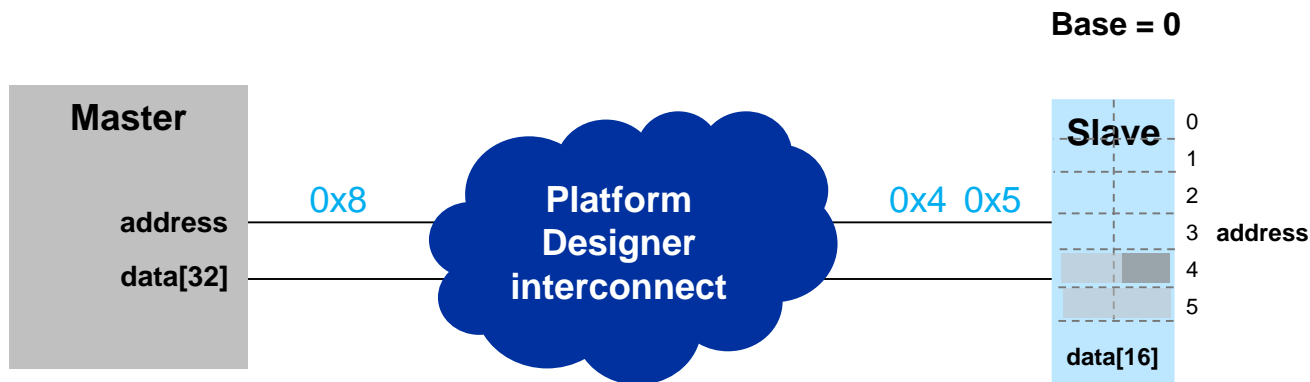
- Пример выравнивания адресов (slave имеет меньшую разрядность)
 - Чтение по адресу Access base + 0x0: dd cc bb aa
 - Система связей выполнит 4 обмена на стороне Ведомого что бы передать Ведущему все 32 бита.
 - Чтение по адресу Access base + 0x4: xx xx xx ee
 - Система связей выполнит 1 обмен на стороне Ведомого, т.к. у ведомого есть только один адресуемый регистр из диапазона адресов.



Регистры Ведомого	
Base	aa
Base + 0x1	bb
Base + 0x2	cc
Base + 0x3	dd
Base + 0x4	ee

Динамическое выравнивание адресов

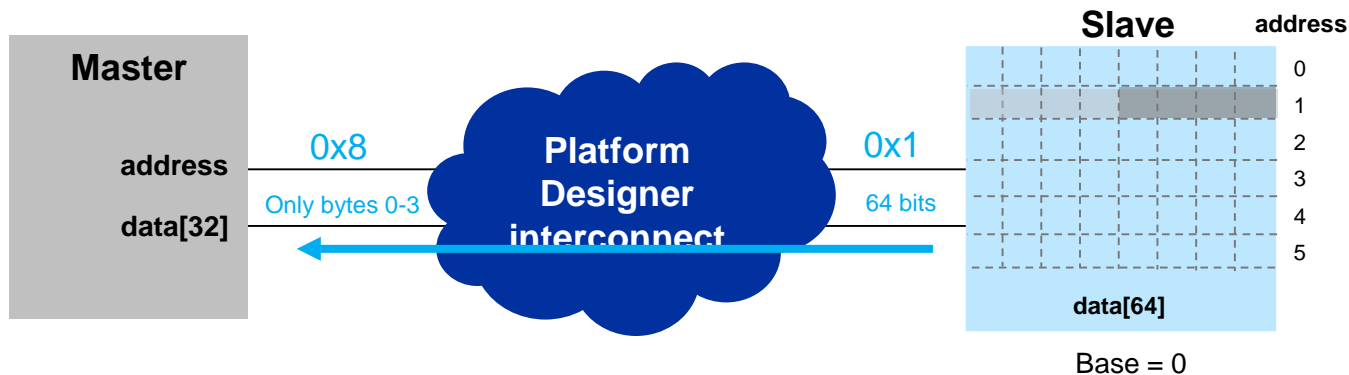
- Пример выравнивания адресов (slave имеет меньшую разрядность)
- Ведомый (Slaves) использует адресацию словами (слово имеет разрядность 16)
- Ведущий (Masters) использует байтовую адресацию (адресует каждый байт)
 - Счетчик адреса в 32-bit Ведущем должен увеличиваться на 4 для доступа к следующему 32-bit слову
 - Чтение по адресу Access base + 0x8:
 - Система связей выполнит 2 обмена на стороне Ведомого что бы передать Ведущему все 32 бита.



Динамическое выравнивание адресов (Wider Slave)

Если Ведомый имеет большую разрядность, то система связей формирует управление `byteenable` для данных от Ведомого, даже если сигналы `byteenable` явно не использованы в интерфейсах

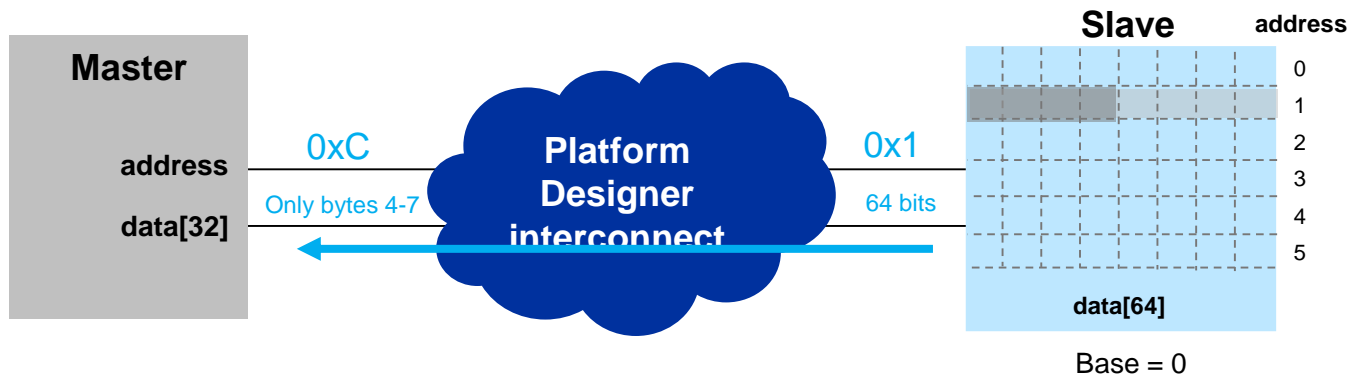
- Обмен Read: Ведущий получает только соответствующий адресу набор байтов
- Обмен Writes: Запись осуществляется только в байты соответствующие адресу, сформированному Ведущим



Динамическое выравнивание адресов (Wider Slave)

Если Ведомый имеет большую разрядность, то система связей формирует управление byteenable для данных от Ведомого, даже если сигналы byteenable явно не использованы в интерфейсах

- Обмен Read: Ведущий получает только соответствующий адресу набор байтов
- Обмен Writes: Запись осуществляется только в байты соответствующие адресу, сформированному Ведущим



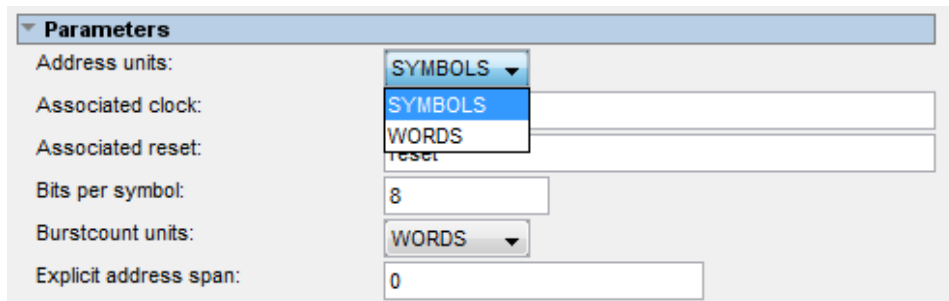
Динамическое выравнивание адресов (Обобщение)

Master Byte Address	Access	32-Bit Master Data		
		When Accessing an 8-Bit Slave Interface	When Accessing a 16-Bit Slave Interface	When Accessing a 64-Bit Slave Interface
0x00	1	OFFSET[0] _{7..0}	OFFSET[0] _{15..0}	OFFSET[0] _{31..0}
	2	OFFSET[1] _{7..0}	OFFSET[1] _{15..0}	—
	3	OFFSET[2] _{7..0}	—	—
	4	OFFSET[3] _{7..0}	—	—
0x04	1	OFFSET[4] _{7..0}	OFFSET[2] _{15..0}	OFFSET[0] _{63..32}
	2	OFFSET[5] _{7..0}	OFFSET[3] _{15..0}	—
	3	OFFSET[6] _{7..0}	—	—
	4	OFFSET[7] _{7..0}	—	—

Адресация Master и Slave

Если Ведомый использует разряды address и реализует адресное декодирование с байтовой адресацией (по младшему разряду адреса) или с доступом к элементам другой разрядности, отличной от разрядности входных данных data, следует настроить параметры интерфейса Slave

- Параметр Address units для slave interface следует установить как SYMBOLS (в Component Editor)
- Параметру Bits per symbol задать соответствующую разрядность (defaults = 8)



The screenshot shows a 'Parameters' dialog box with the following settings:

Parameter	Value
Address units:	SYMBOLS
Associated clock:	SYMBOLS
Associated reset:	reset
Bits per symbol:	8
Burstcount units:	WORDS
Explicit address span:	0

- Для Ведущего также можно изменить Address units на word

Ведомый по умолчанию (Default Slave)

Запрос Ведущего перенаправляется к “default slave” если:

- В адресном пространстве Ведущего есть зазоры
- Запрос по адресу вне адресного пространства Ведущего
- Запрос на запись к Read-only Ведомому (запрос на Чтение к write-only Ведомому)

- Назначается Ведомый для формирования ответов readdata и/или response
 - Если явно не задан такой Ведомый, то им по умолчанию будет Ведомый с меньшим адресом в адресном пространстве Ведущего

Name	Description	Export	Clock	Base	End	Default Slave
RGB_DATA	On-Chip Memory (RAM or ROM)					
clk1	Clock Input	Double-click to export	pll_outclk0			
reset1	Reset Input	Double-click to export	[clk1]			
s1	Avalon Memory Mapped Slave	Double-click to export	[clk1]	default	default	<input checked="" type="checkbox"/>

Данная колонка не отображена по умолчанию.
right-click в области заголовков колонок и выберите Show default Slave Column

- ✓ Show Use Column
- ✓ Show Connections Column
- ✓ Show Description Column
- ✓ Show Clock Column
- ✓ Show Base Column
- ✓ Show End Column
- ✓ Show Export Column
- Show Auto Export Column
- ✓ Show Tags Column
- Show Security Column
- ✓ Show Default Slave Column
- ✓ Show Opcode Name Column
- ✓ Show IRQ Column

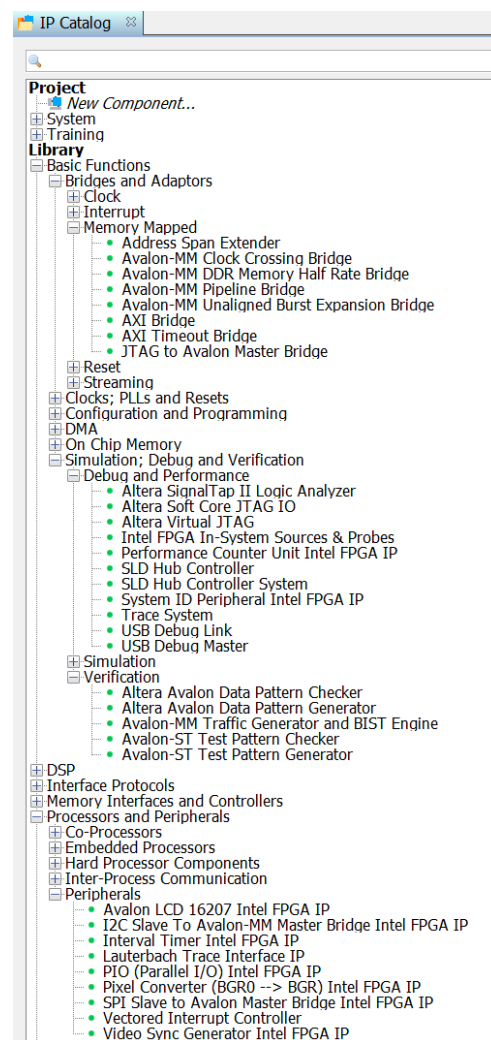
План

- Интерфейсы шины Avalon-MM
- Типы обменов на шине Avalon-MM
- Адресация на шине Avalon-MM
- Компоненты шины Avalon-MM
- Упражнение 3

Компоненты Avalon-MM

Используются для реализации управления (control pane) и передачи данных (data pane):

- On-Chip Memory (RAM and ROM) Core
- PIO Core
- PLL Cores
- Bridges....:
 - Адаптеры могут добавляться либо автоматически, либо вручную.
- DMA Controller Core
-

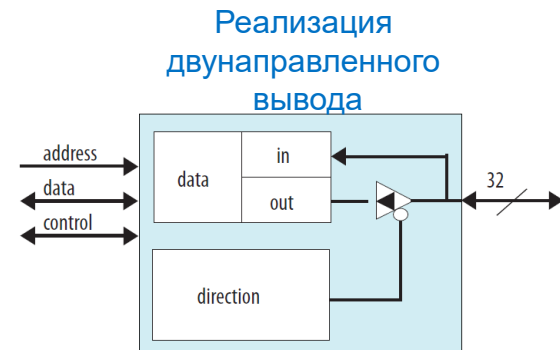
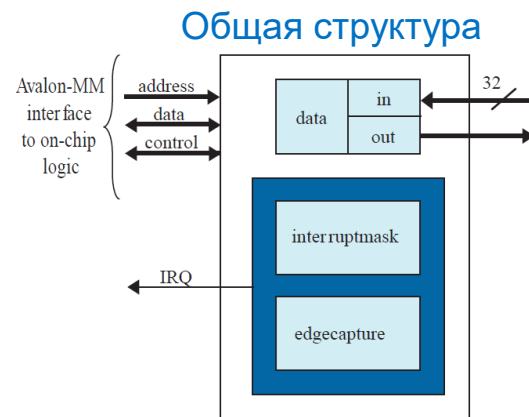


Компонент PIO Core

Обеспечивает интерфейс между Avalon-MM и портами ввода-вывода:

- От 1 до 32 выводов
- 4 регистра: data, direction, interruptmask, edgecapture
- Карта памяти

Offset	Register Name		R/W
0	data	read access	R
		write access	W
1	direction		R/W
2	interruptmask		R/W
3	edgecapture		R/W
4	outset		W
5	outclear		W



Компонент PIO Core: настройка

PD позволяет настроить PIO для конкретного применения

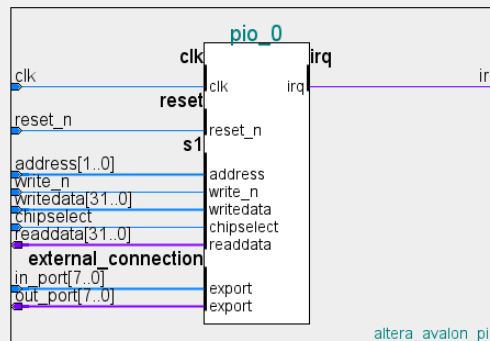
PIO (Parallel I/O) Intel FPGA IP - pio_0



PIO (Parallel I/O) Intel FPGA IP
altera_avalon_pio

Block Diagram

☒ Show signals



Basic Settings

Width (1-32 bits):

Direction:

- ☐ Bidir
- ☐ Input
- ☒ InOut
- ☐ Output

Output Port Reset Value:

Output Register

☐ Enable individual bit setting/clearing

Edge capture register

☒ Synchronously capture

Edge Type:

☒ Enable bit-clearing for edge capture register

Interrupt

☒ Generate IRQ

IRQ Type:

Level: Interrupt CPU when any unmasked I/O pin is logic true
Edge: Interrupt CPU when any unmasked bit in the edge-capture register is logic true. Available when synchronous capture is enabled

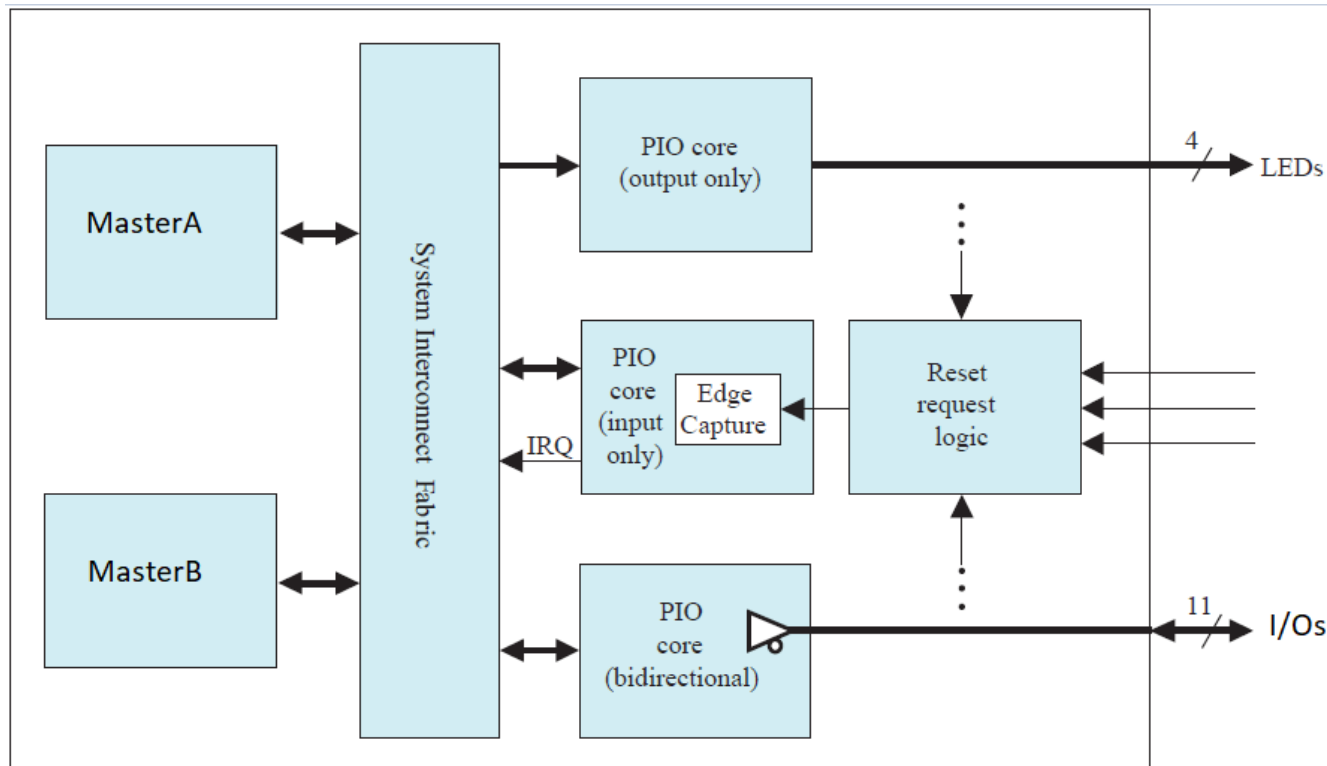
Test bench wiring

☐ Hardwire PIO inputs in test bench

Drive inputs to field.:

Компонент PIO Core: пример системы

Реализация обмена данным с внешними выводами

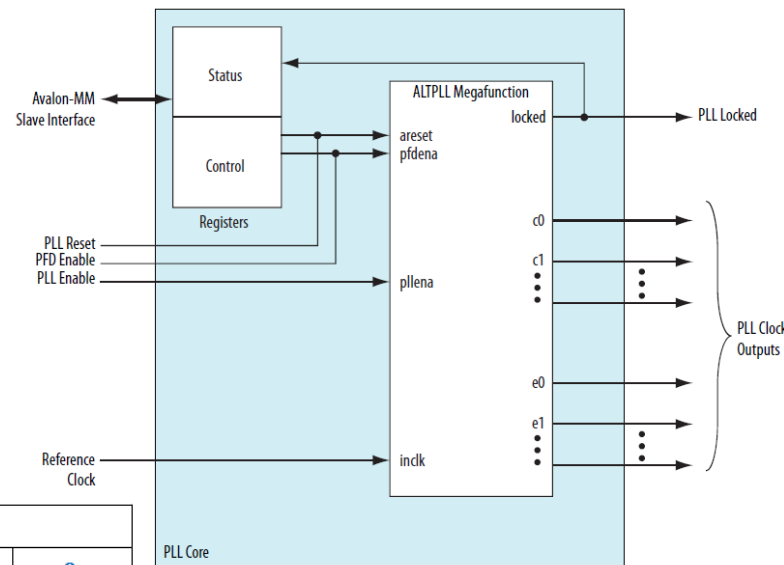


Компонент Avalon ALTPLL Core

Общая структура

Формирование тактовых сигналов на базе
ВХОДНОГО ТАКОВОГО СИГНАЛА

- 2 регистра: status, control, phase reconfig control
- Карта памяти:



Offset	Register Name	R/W	Bit Description												
			31	30	29	...	9	8	7	6	5	4	3	2	1
0	status	R/O	Undefined											phasedone	locked
1	control	R/W	Undefined											pfdena	areset
2	phase reconfig control	R/W	phase		Undefined			counter_number							
3	—	—	Undefined												

Компонент Avalon ALTPLL Core: настройка

PD позволяет настроить режимы ALTPLL Core

MegaWizard Plug-In Manager [page 1 of 11]

ALTPLL

1 Parameter Settings 2 PLL Reconfiguration 3 Output Clocks 4 EDA

General/Modes Inputs/Lock Bandwidth/SS Clock switchover

ALTPLL1618375417715218

Currently selected device family: Cyclone IV E

☒ Match project/default

General

Which device speed grade will you be using? Any

☐ Use military temperature range devices only

What is the frequency of the inclk0 input? 100.000 MHz

☐ Set up PLL in LVDS mode Data rate: Not Available Mbps

PLL Type

Which PLL type will you be using?

☐ Fast PLL ☐ Enhanced PLL ☒ Select the PLL type automatically

Operation Mode

How will the PLL outputs be generated?

☒ Use the feedback path inside the PLL

☒ In normal mode

☐ In coarse-synchronous compensation mode

☐ In zero delay buffer mode

☐ Connect the trimmable port (bidirectional)

☐ With no compensation

☐ Create an 'tbin' input for an external feedback (External Feedback Mode)

Which output clock will be compensated for? c0

Cancel < Back Next > Finish

1 Parameter Settings 2 PLL Reconfiguration 3 Output Clocks 4 EDA

General/Modes Inputs/Lock Bandwidth/SS Clock switchover

ALTPLL1618375417715218

inclk0 areset scandataout scandone locked

inclk0 frequency: 100.000 MHz

Operation Mode: Normal

clk Ratio Ph (deg) DC (%)

c0 1/1 0.00 50.00

Cyclone IV E

able to implement the requested PLL

Optional Inputs

☐ Create an 'plena' input to selectively enable the PLL

☒ Create an 'areset' input to asynchronously reset the PLL

☒ Create an 'pfdena' input to selectively enable the phase/frequency detector

Lock Output

☒ Create 'locked' output

☐ Enable self-reset on loss lock

Advanced Parameters

Using these parameters is recommended for advanced users only

☐ Create output file(s) using the 'Advanced' PLL parameters

- Configurations with output clock(s) that use cascade counters are not supported

Avalon Bus connectivity

☐ Use a separate clock input for Avalon bus connections

1 Parameter Settings 2 PLL Reconfiguration 3 Output Clocks 4 EDA

clk c0 clk c1 clk c2 clk c3 clk c4

ALTPLL1618375417715218

inclk0 areset c0 c1 c2 c3 c4 scandataout scandone locked

inclk0 frequency: 100.000 MHz

Operation Mode: Normal

clk Ratio Ph (deg) DC (%)

c0 1/1 0.00 50.00

c1 2/1 0.00 50.00

c2 3/1 0.00 50.00

c3 1/1 0.00 50.00

c4 3/7 0.00 50.00

Cyclone IV E

c4 - Core/External Output Clock

☒ Use this clock

Clock Tap Settings

☐ Enter output clock frequency

☒ Enter output clock parameters

Requested Settings Actual Setting

0.00000000 MHz 2.857143

Enter output clock parameters

Clock multiplication factor 3 3

Clock division factor 7 << Copy 7

Clock phase shift 0.00 deg 0.00

Clock duty cycle (%) 50.00 50.00

Note: The displayed internal settings of the PLL is recommended for use by advanced users only

Description Primary clock VCO frequency... 6


Per Clock Feasibility Indicator c0 c1 c2 c3 c4

Компонент Avalon ALTPLL Core: настройка

PD позволяет подготовить файлы настройки для динамической реконфигурации

MegaWizard Plug-In Manager [page 5 of 11]

?

 **ALTPLL**

1 Parameter Settings

2 PLL Reconfiguration

3 Output Clocks

4 EDA

ALTPLL1618375417715218

inclk0

areset

scanclk

scandata

scanclkena

configupdate

inclk0 frequency: 100.000 MHz

Operation Mode: Normal

Clk	Ratio	Ph (dg)	DC (%)
c0	1/1	0.00	50.00

c0

scandataout

scandone

locked

Cyclone IV E

Dynamic Reconfiguration

☒ Create optional inputs for dynamic reconfiguration
Used for non-phase (e.g. frequency, duty cycle, bandwidth, etc.) reconfiguration
- Note: Reconfiguration with cascaded counters may not work correctly

Initial Configuration File
Use the following initial configuration file to initialize the altpll_reconfig megafunction (Valid file formats are the Hexadecimal (Intel-format) [.hex] and the Memory Initialization File
File name:

Additional Configuration File
You may create additional configuration file(s) for the current PLL settings. These files may be used to initialize the altpll_reconfig megafunction.
To create a configuration file, enter a valid file name and press the 'Generate A Configuration File' button (Valid file formats are the Hexadecimal (Intel-format) [.hex] and the Memory Initialization File [.mif]).
File name:

Reconfiguration file generation was successful

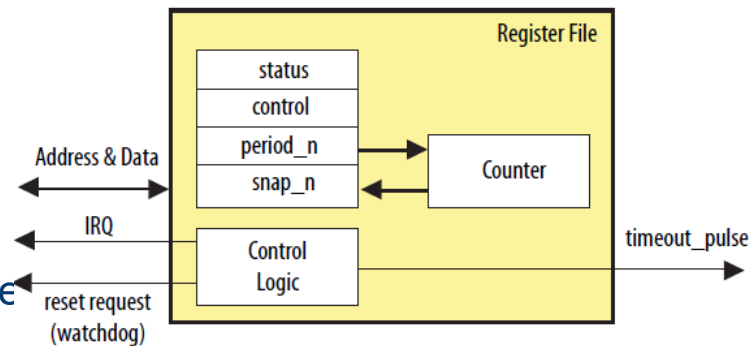
Dynamic Phase Reconfiguration
☐ Enable phase shift step resolution
☐ Create optional inputs for dynamic phase reconfiguration

Компонент Interval Timer Core

Общая структура

Программируемый интервальный таймер:

- От 32 или 64 разряда
- Управление: старт, стоп, продолжить.
- Фиксация snapshot – запись в регистр (записываемые данные игнорируются)
- Карта памяти (32 битный регистр)



Offset	Name	R/W	Description of Bits						
			15	...	4	3	2	1	0
0	status	RW						RUN	TO
1	control	RW				STOP	START	CONT	ITO
2	periodl	RW	Timeout Period – 1 (bits [15:0])						
3	periodh	RW	Timeout Period – 1 (bits [31:16])						
4	snaph	RW	Counter Snapshot (bits [15:0])						
5	snaph	RW	Counter Snapshot (bits [31:16])						

Компонент Interval Timer Core : настройка

PD позволяет настроить режимы и выбрать единицу измерения периода

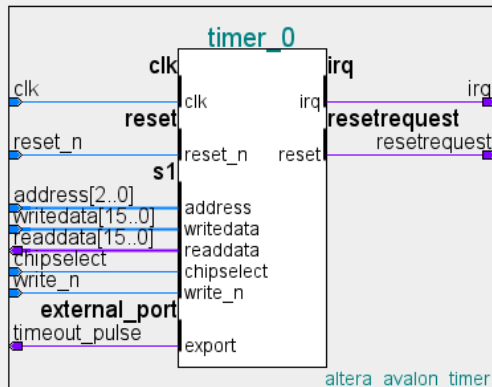
Interval Timer Intel FPGA IP - timer_0



Interval Timer Intel FPGA IP
altera_avalon_timer

Block Diagram

☒ Show signals



Timeout period

Period:

1

Units:

clocks

Timer counter size

Counter Size:

32

Units (periodUnits):

Timeout Period setting can be specified in units of Ojs, ms, seconds, or clocks.

Registers

☐ No Start/Stop control bits

☒ Fixed period

☒ Readable snapshot

Output signals

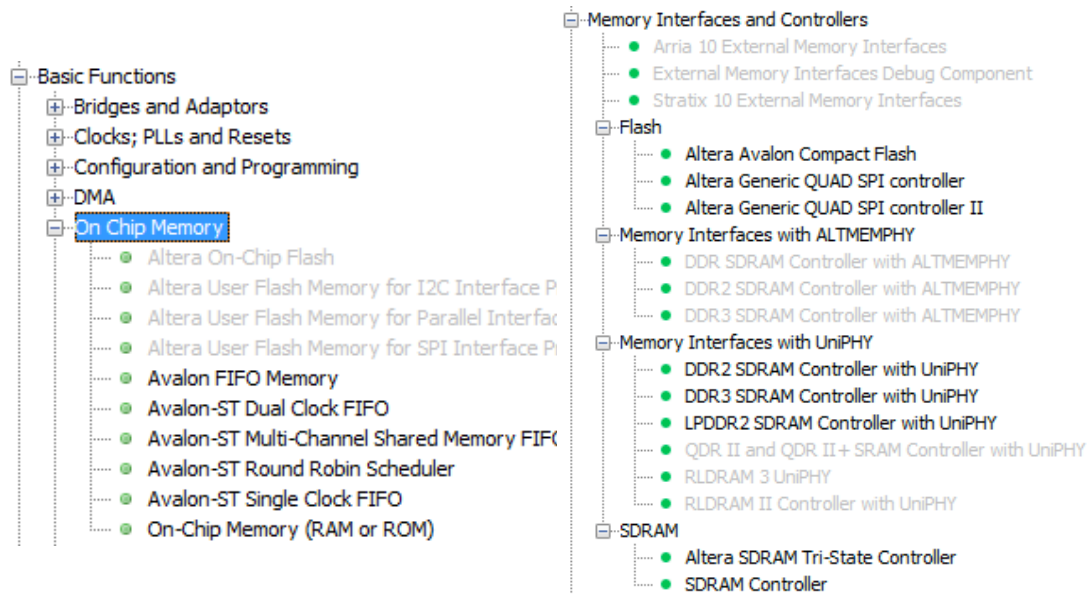
☒ System reset on timeout (Watchdog)

Watchdog Timer Pulse Length: 2

☒ Timeout pulse (1 clock wide)

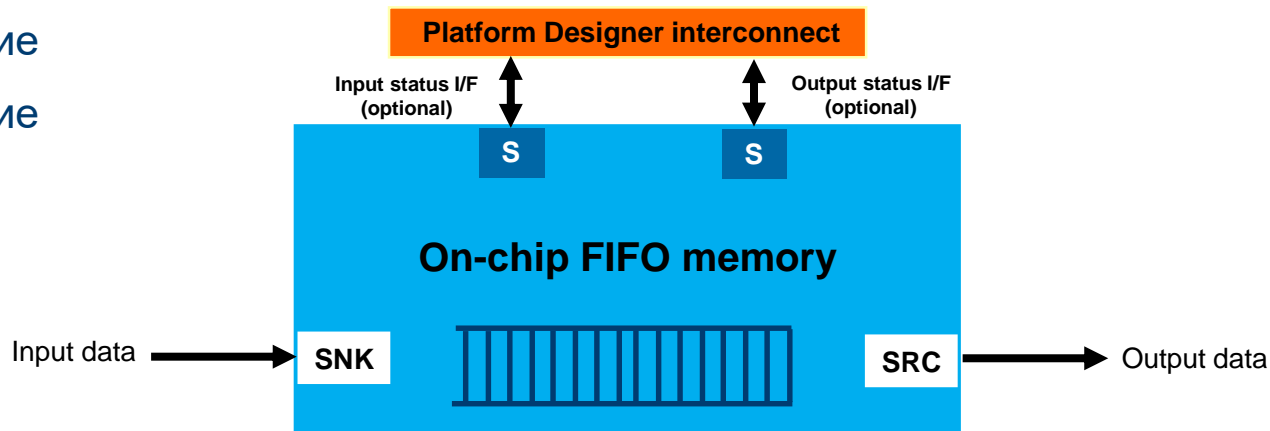
Компоненты памяти: Memory IP

- On-chip
 - On-chip RAM/ROM
 - On-chip FIFO
- Off-chip
 - SDRAM Controller
 - DDR controllers
 - QDR II/QDR II+ SRAM controllers
 - RLDRAM II/3 controllers
 - LPDDR2 controller
 - Flash interfaces



Пример: On-Chip FIFO

- Буферизирует данные: первый пришел – первый ушел
- Один и два тактовых сигнала
- Поддержка Avalon-MM и/или Avalon-ST интерфейсы
 - MM для записи и чтения
 - ST для записи и чтения
 - MM запись - ST чтение
 - ST запись - MM чтение

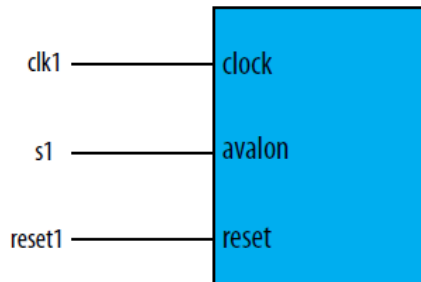


Компонент On-Chip Memory (RAM and ROM)

Позволяет использовать блоки
встроенной памяти FPGA

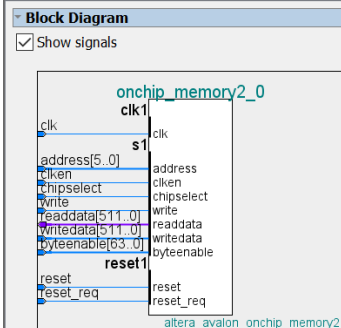
- Разрядность: 8, 16, 32, 64, 128, 256, 512, 1024 разряда

Общая структура



On-Chip Memory (RAM or ROM) Intel FPGA IP - onchip_memory2_0

On-Chip Memory (RAM or ROM) Intel FPGA IP
altera_avalon_onchip_memory2



Block Diagram

☒ Show signals

Memory type

Type:

RAM (Writable)

☐ Dual-port access

☐ Single clock operation

Read During Write Mode:

DONT_CARE

Block type:

AUTO

Size

☐ Enable different width for Dual-port access

Slave s1 Data width:

512

Total memory size:

4096

bytes

☐ Minimize memory block usage (may impact fmax)

Read latency

Slave s1 Latency:

1

Slave s2 Latency:

1

ROM/RAM Memory Protection

Reset Request:

Enabled

ECC Parameter

Extend the data width to support ECC bits:

Disabled

Memory initialization

☒ Initialize memory content

☒ Enable non-default initialization file

Type the filename (e.g. my_ram.hex) or select the hex file using the file browser button.

User created initialization file:

onchip_mem.hex

☐ Enable Partial Reconfiguration Initialization Mode

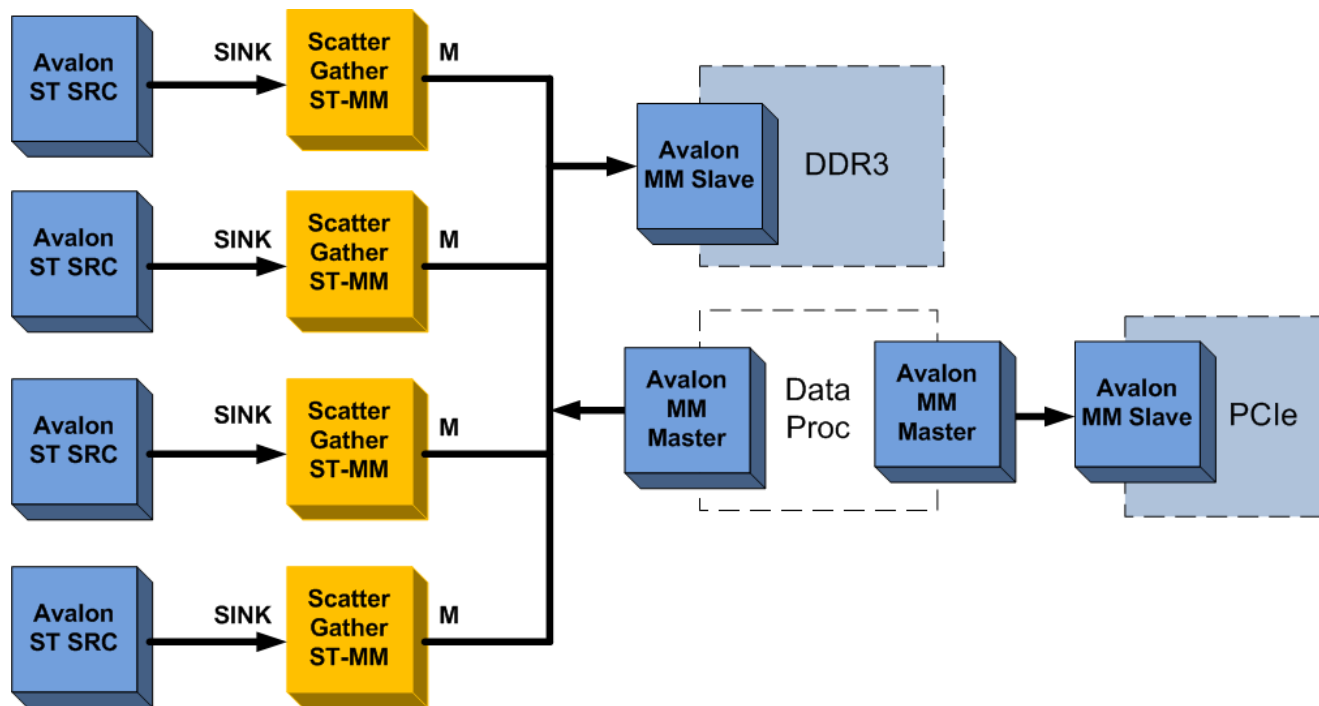
☐ Enable In-System Memory Content Editor feature

Instance ID:

NONE

User is required to provide memory initialization files for memory.
Memory will be initialized from onchip_mem.hex

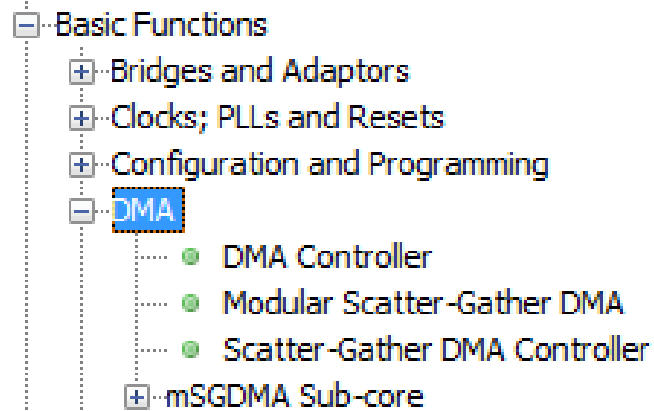
Пример системы



Компоненты Direct Memory Access (DMA)

- DMA
 - Perform bulk data transfers between contiguous Avalon[®]-MM interface address ranges
- Scatter-Gather DMA
 - Perform multiple bulk data transfers between non-contiguous memory and continuous address space
 - Read/write between MM and ST interfaces
- Modular SGDMA
 - Similar to SGDMA, but separate read, write, and dispatcher modules for more flexibility

Library



Компонент Scatter-Gather DMA

Обеспечивает передачу данных без участия процессора (или другого Ведущего)
Descriptors

- Конфигурации
 - Memory-Mapped to Memory-Mapped
 - Memory-Mapped to Streaming
 - Streaming to Memory-Mapped
- Карта памяти

Offset	Access	Byte Lanes			
		3	2	1	0
0x0	Write	Read Address[31:0]			
0x4	Write	Write Address[31:0]			
0x8	Write	Length[31:0]			
0xC	Write	Control[31:0]			

CSR

Offset	Attribute	Byte Lanes			
		3	2	1	0
0x0	Read/Clear	Status			
0x4	Read/Write	Control			
0x8	Read	Write Fill Level[15:0]		Read Fill Level[15:0]	
0xC	Read	<reserved>		Response Fill Level[15:0]	
0x10	Read	Write Sequence Number[15:0] ⁽²⁹⁾		Read Sequence Number[15:0]	

Компонент Scatter-Gather DMA : настройка

PD позволяет выбрать режим и настроить его особенности

Scatter-Gather DMA Controller Intel FPGA IP - sgdma_0

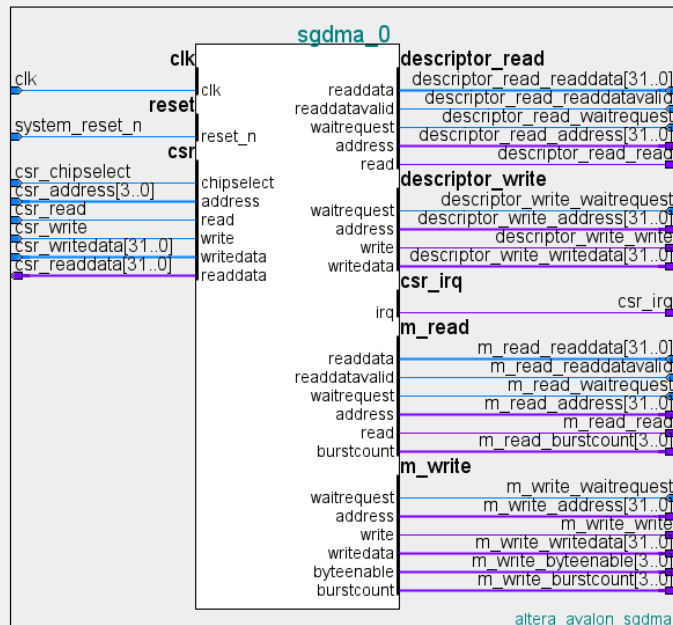


Scatter-Gather DMA Controller Intel FPGA IP
altera_avalon_sgdma

Documentation

Block Diagram

☒ Show signals



Transfer options

Transfer mode:

Memory To Memory

☐ Enable bursting on descriptor read master

Memory To Memory

☐ Allow unaligned transfers

Memory To Stream

Stream To Memory

Transfer mode (transferMode):
Configuration of Transfer mode.

☒ Enable burst transfers

Read burstcount signal width:

4

Write burstcount signal width:

4

Avalon MM data master byte reorder mode:

No Reordering

Data and error widths

Data width:

32

Source error width:

0

Sink error width:

0

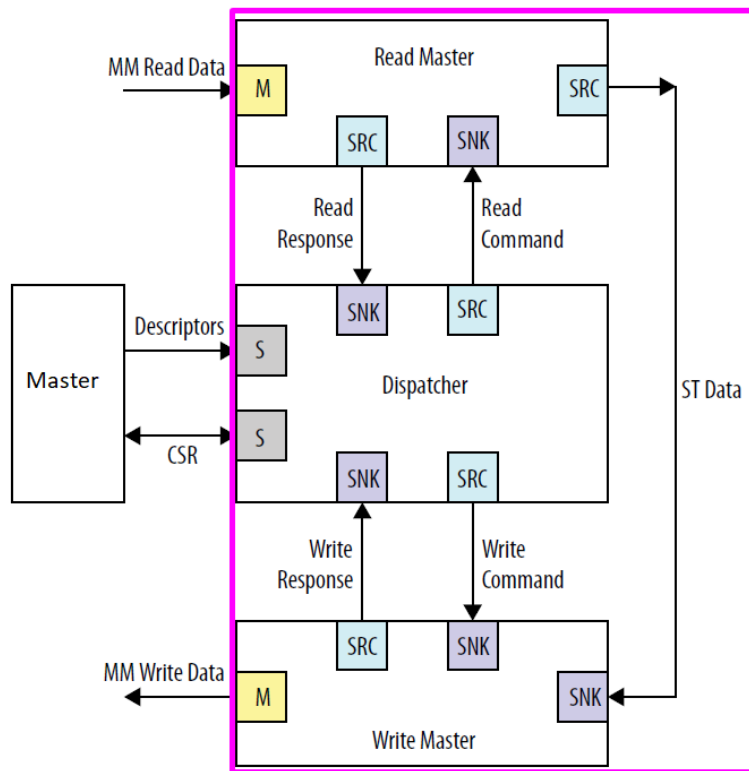
FIFO depth

Data transfer FIFO depth:

64

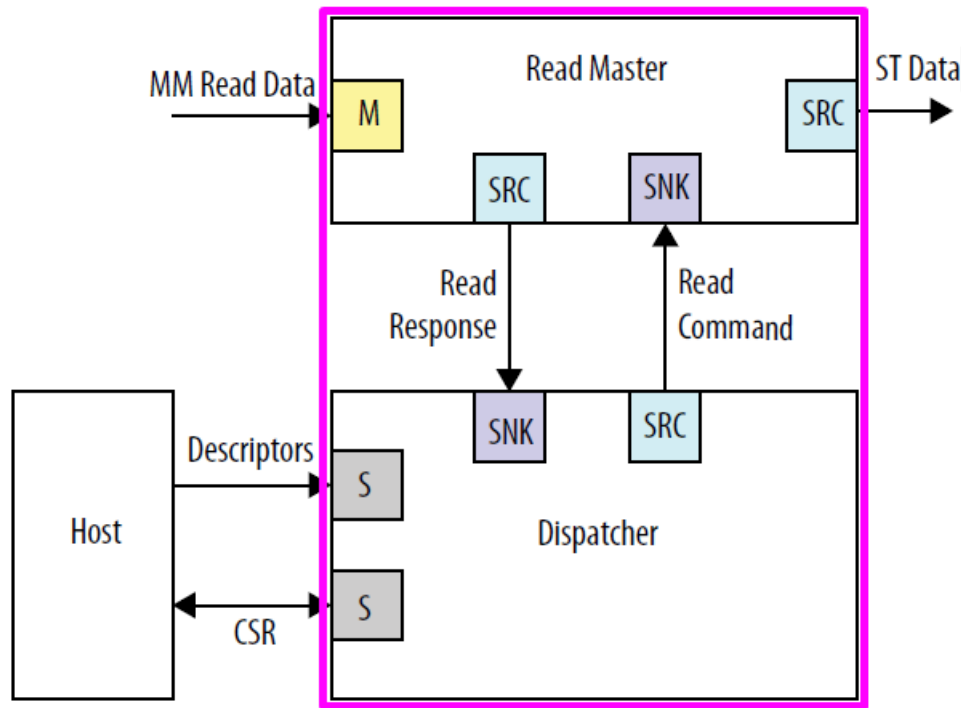
Компонент mScatter-Gather DMA: пример системы

Реализация обмена данным Memory-Mapped to Memory-Mapped



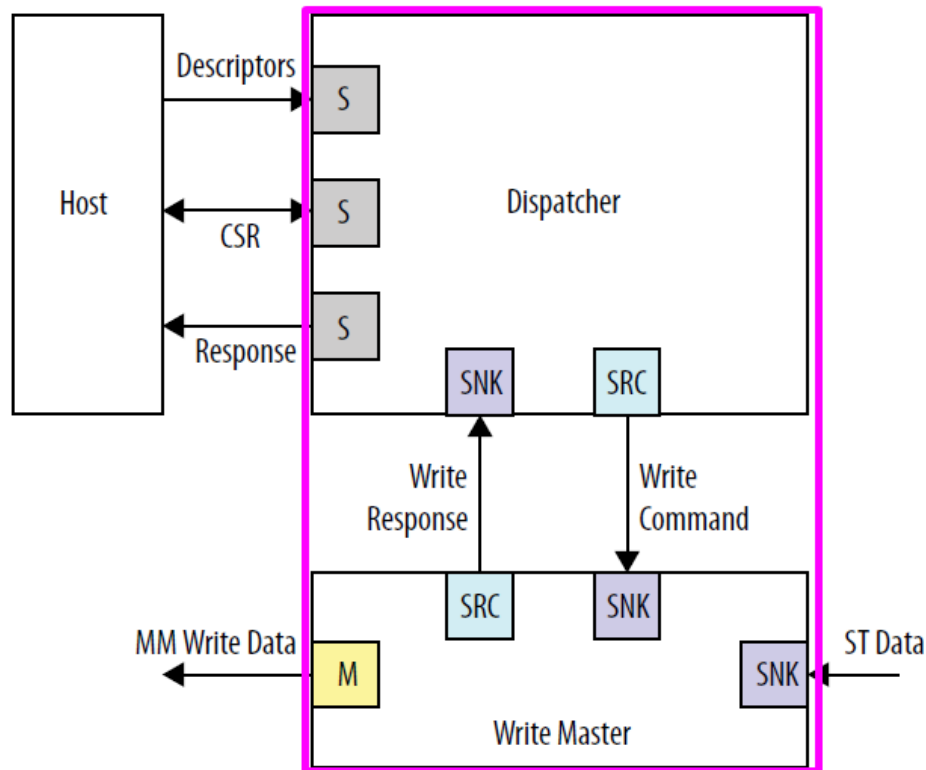
Компонент mScatter-Gather DMA: пример системы

Реализация обмена данным Memory-Mapped to Streaming



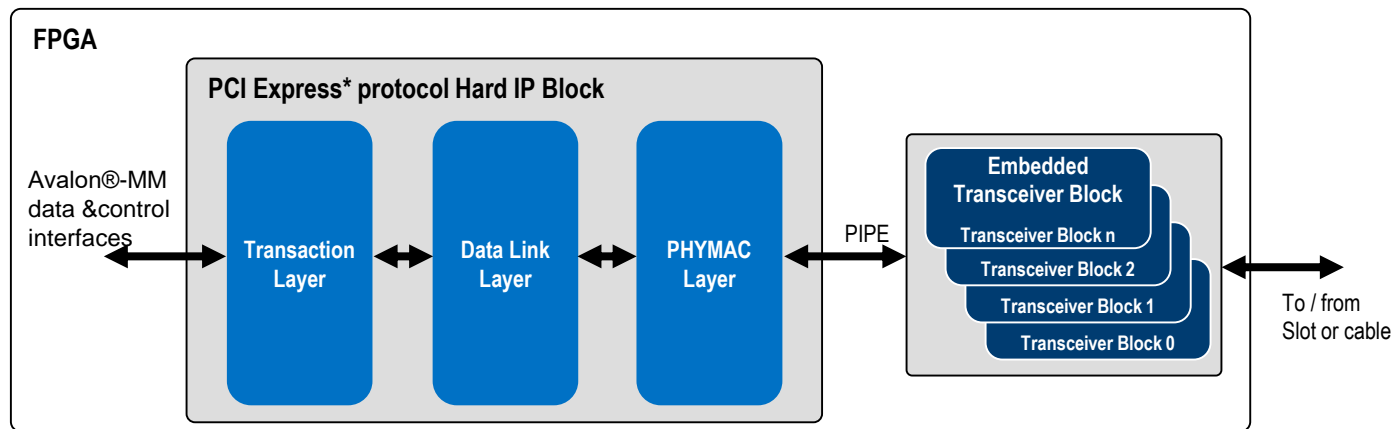
Компонент mScatter-Gather DMA: пример системы

Реализация обмена данным Streaming to Memory-Mapped



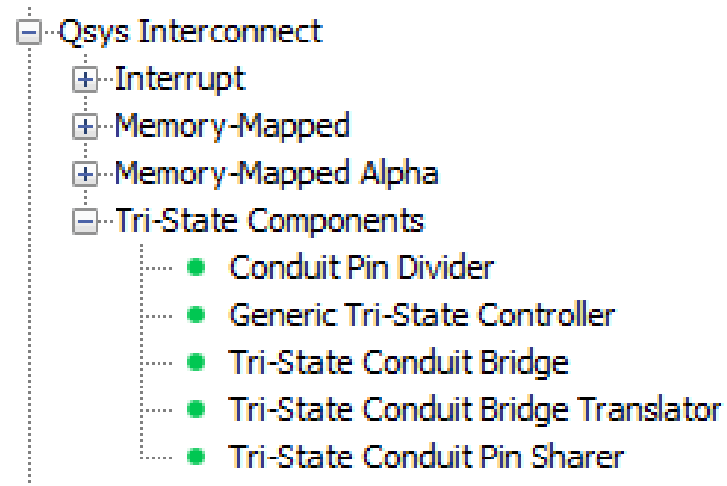
Пример: Hard IP for PCI Express* protocol

- Конфигурация Hard IP for PCI Express* с использованием встроенных трансиверов
 - Реализация уровней: transaction, data link, PHYMAC layer
 - Поддержка: PCI Express Gen 3 (8.0 Gbps), Gen 2 (5.0 Gbps), & Gen 1 (2.5 Gbps)
 - Поддержка режимов: root port и endpoint
- Подключение к Avalon®-MM interface master и slave

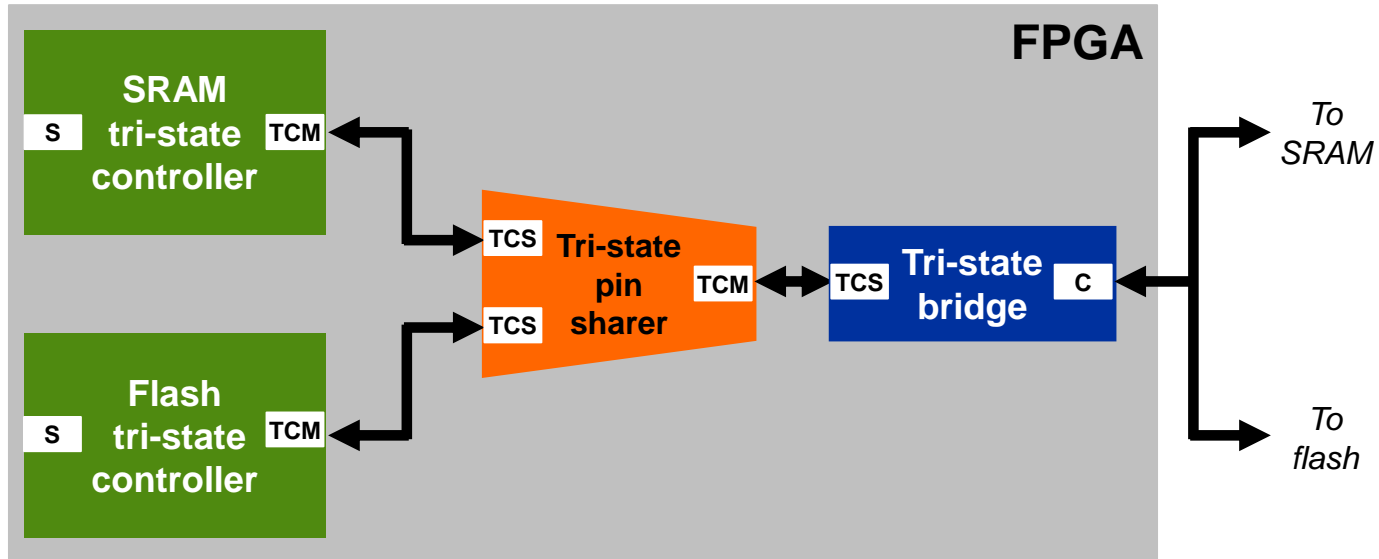


Компоненты Tri-State IP

- Generic Tri-State Controller
 - Bridge between Avalon® interface slave and tri-state conduit master (TCM) interfaces for controlling external components on a tri-state bus
- Tri-State Pin Sharer
 - Mux to allow off-chip devices to share FPGA pins
 - Example: same address pins used for each device
- Tri-State Conduit Bridge
 - Bridges unidirectional core signals and tri-state bidirectional I/O signals
 - Must be used to connect to off-chip tri-state devices



Пример использования Tri-State Component



TCM - Tri-state conduit master interface

TCS - Tri-state conduit slave interface

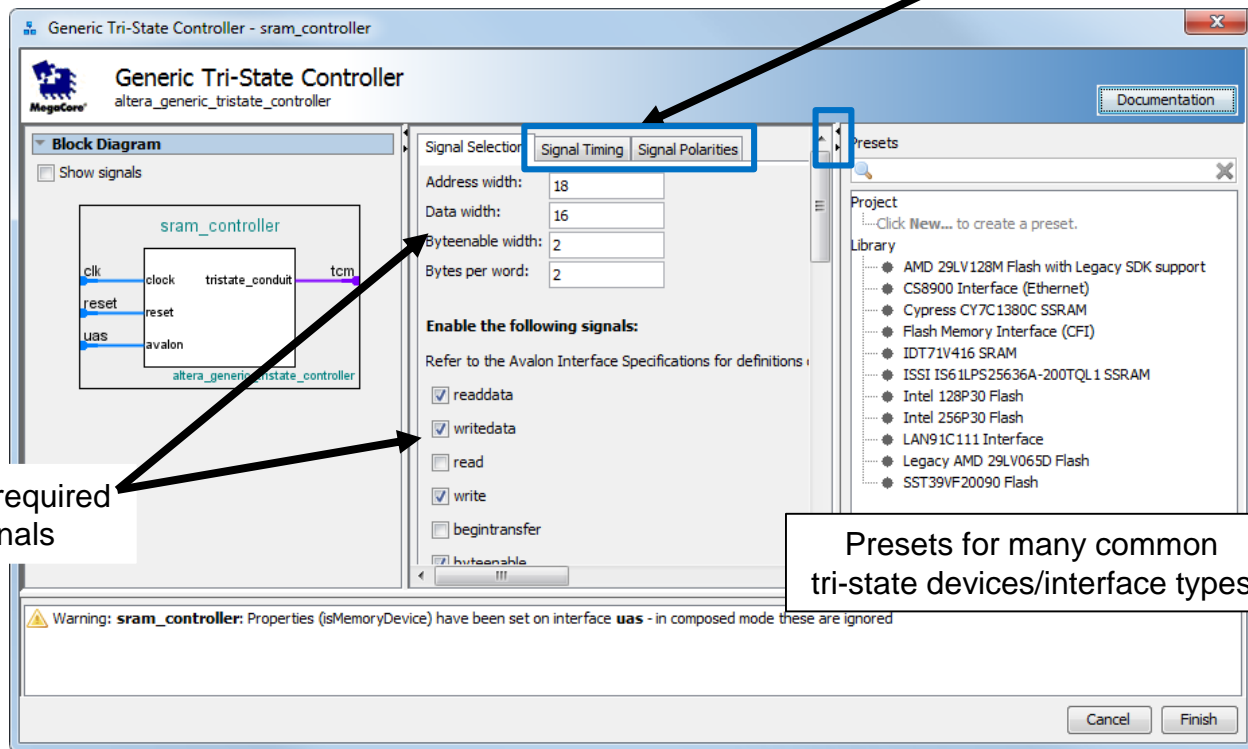
C - Conduit interface

Настройка Tri-State Controller

Define device-specified signal timing and polarity
(e.g. wait, setup, & hold cycles)

Bus widths and required
interface signals

Presets for many common
tri-state devices/interface types



Настройка Tri-State Conduit Pin Sharer

of tri-state conduits to be shared

If conduits already connected, auto-fill table

Update Interface Table

Interface	Signal Role	Signal Type	Signal Width	Shared Signal Name
sram_controller.tcm	address	Output	20	addr
sram_controller.tcm	outputenable_n	Output	1	oe
sram_controller.tcm	byteenable_n	Output	4	be
sram_controller.tcm	begintransfer_n	Output	1	
sram_controller.tcm	write_n	Output	1	write_n
sram_controller.tcm	data	Bidirectional	32	data

Add/remove signals to be shared

If same name used for signals from separate conduits, signal is shared

Block Diagram

tristate_conduit_pin_sharer_0

Parameters

Number of Interfaces: 2

Sharing Assignment

To share a signal, type the same signal name in the Shared Signal Name column for all controllers that share that signal

Update Interface Table

Interface

Signal Role

Signal Type

Signal Width

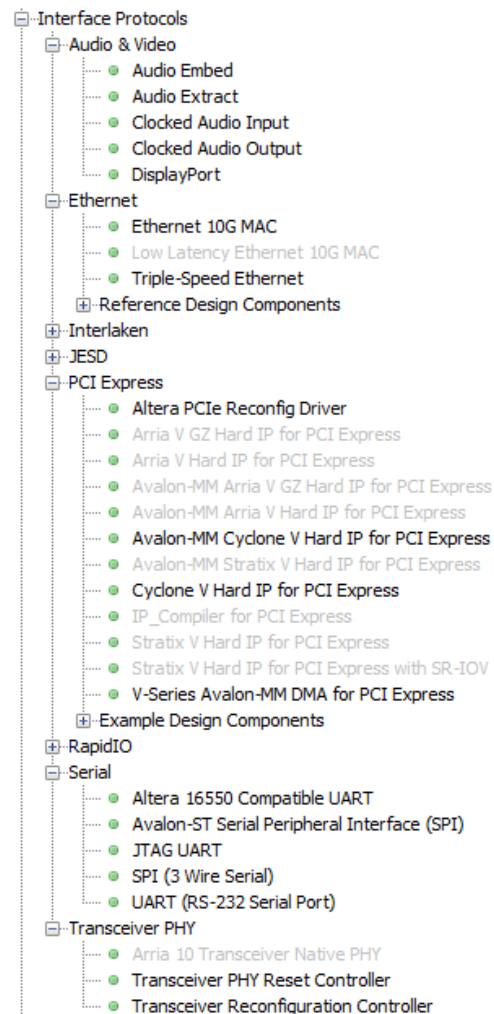
Shared Signal Name

+

-

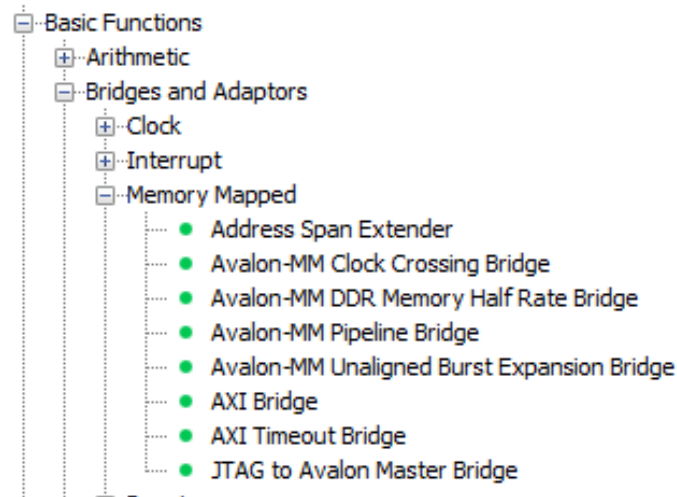
Компоненты High-Speed Interface IP

- 10/100/1000 Mb (Triple-Speed) Ethernet
- PCI Express* protocol
- 10 Gb Ethernet
- Interlaken
- RapidIO*
- ...



Компоненты Memory-Mapped Bridge IP

- Avalon®-MM interface Clock Crossing bridge
 - FIFOs for buffered high-throughput clock domain crossing between master and slave
- Avalon-MM Pipeline Bridge
 - Control interface topology with or without pipeline latency
 - Allow exporting of multiple MM interfaces through one aggregate interface
- Unaligned Burst Expansion Bridge
 - Improves performance when a master performs a burst read at an unaligned address location on the slave
- JTAG to Avalon interface Master Bridge
 - Use JTAG commands to access and control via components' slave interfaces
 - Usually used with System Console debugging tool and Bus Analyzer



План

- Интерфейсы шины Avalon-MM
- Типы обменов на шине Avalon-MM
- Адресация на шине Avalon-MM
- Компоненты шины Avalon-MM
- Лабораторная 3

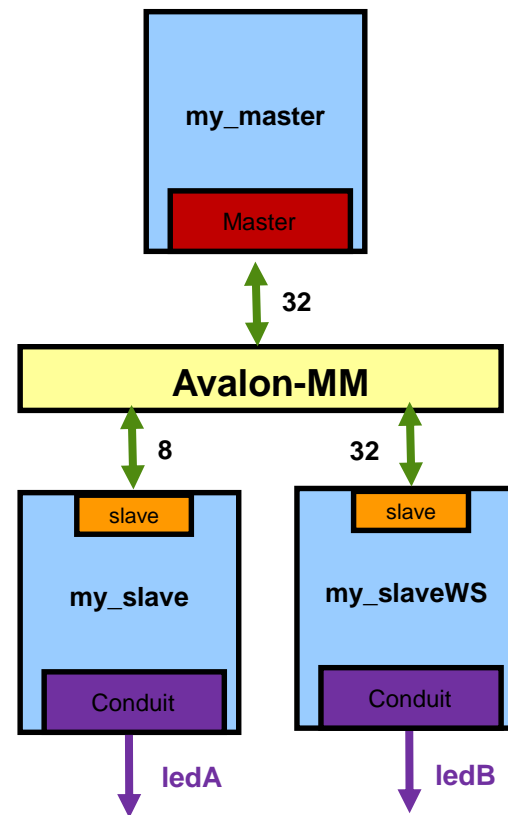


Лабораторная 3

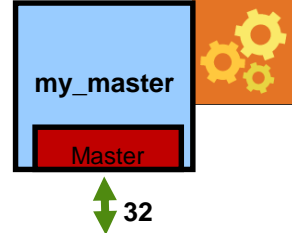


Структура проекта

- 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 waitrequest от системы соединений).
- Ведомый my_slaveWS – 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.



Структура проекта: my_master

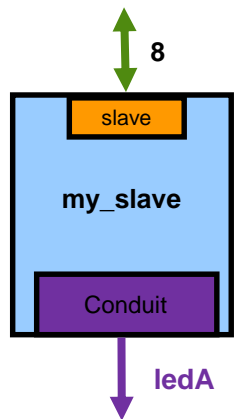


```
1 `timescale 1 ns / 1 ns
2 module my_master #(
3     parameter [31:0] address_1 = 0, parameter [31:0] data_1 = 1,
4     parameter [31:0] address_2 = 1, parameter [31:0] data_2 = 16383
5 ) (
6     // clock and reset
7     input bit csi_clk, // clock clk
8     input bit rsi_reset, // reset reset
9     // MM Master
10    output bit [31:0] avm_m0_address, // MM Master address
11    output bit avm_m0_write, // MM Master write
12    output bit [31:0] avm_m0_writedata, // MM Master writedata
13    input bit avm_m0_waitrequest // MM Master waitrequest
14 );
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;
```

```
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
```



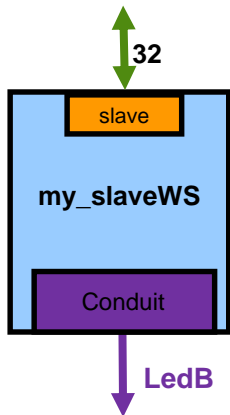
Структура проекта: my_slave



```
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
```



Структура проекта: my_slaveWS



```
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
```



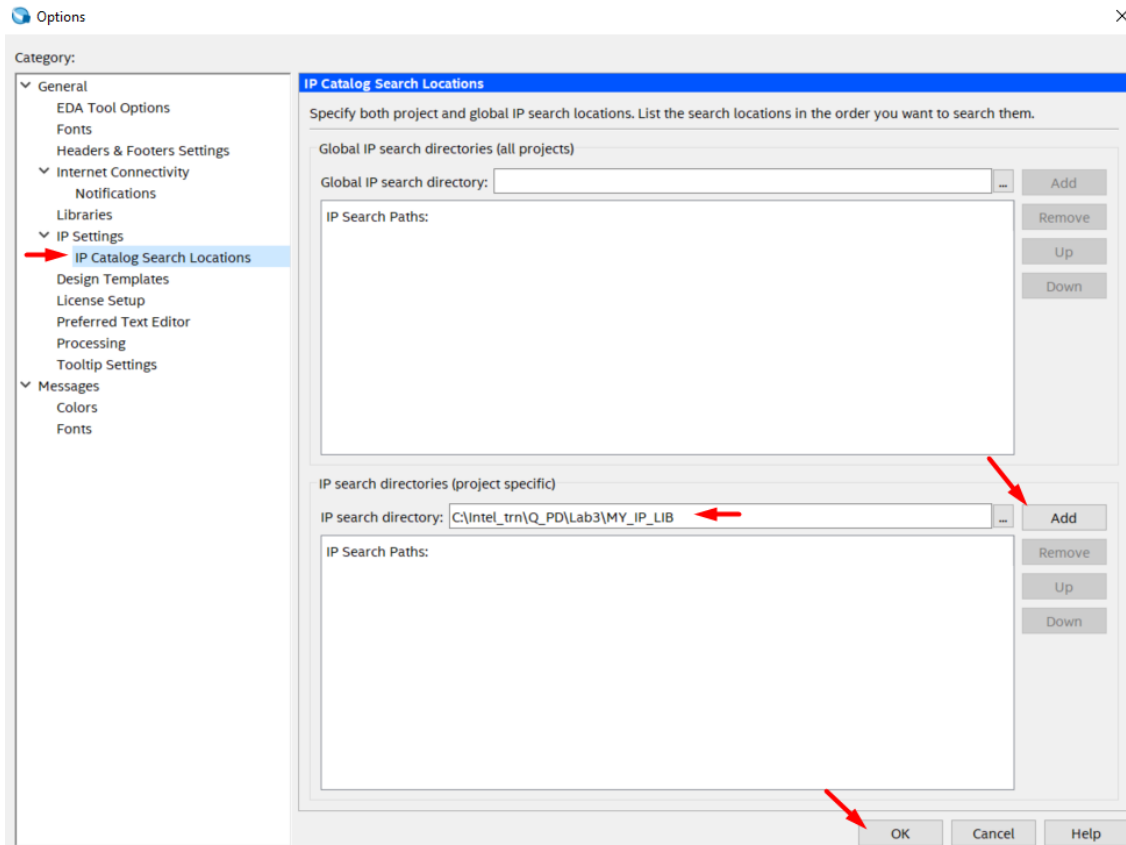
В QP создайте проект

- **Рабочая папка:** C:\Intel_trn\Q_PD\Lab3
- **Имя проекта:** Lab3
- **Модуль верхнего уровня:** Lab3
- **Тип проекта:** Empty Project
- **Файлы не добавляются**
- **Микросхема:** может быть любой
 - Плата DE1-SOC - 5CSEMA5F31C6N
 - Плата SoC Kit - 5CSXFC6D6F31
 - Плата MAX10_NEEK - 10M50DAF484C6G
 - Плата miniDilabCIV (**выбирается по умолчанию**) - EP4CE6E22C8
- **EDA Tool Settings:** Simulation => ModelSim Altera Starter Edition



В QP задайте путь к библиотеке IP

- Команда: Tools=>Options





В QP запустите приложение PD

- **Команда:** Tools => Platform Designer или иконка 
- **В PD:** сохраните систему под именем **Lab3_1.qsys** в рабочей папке проекта
- Убедитесь, что Ваша система выглядит так же, как показано на рисунке ниже

Platform Designer - Lab3_1.qsys (C:\Intel_trn\Q_PD\Lab3\Lab3_1.qsys)

File Edit System Generate View Tools Help

IP Catalog System Contents Address Map Interconnect Requirements

System: Lab3_1

Use	Con...	Name	Description	Export
<input checked="" type="checkbox"/>		clk_0	Clock Source	clk
		clk_in	Clock Input	reset
		clk_in_reset	Reset Input	
		clk	Clock Output	
		clk_reset	Reset Output	

Project

- New Component...
- System
 - Lab3_1
- Training
 - my_master
 - my_slave
 - my_slaveWS

Library

- Basic Functions
- DSP
- Interface Protocols
- Low Power

New... Edit... Add...

Hier Device

- Lab3_1 [Lab3_1.qsys]
- clk
- reset
- clk_0

Messages

Type	Path	Message
------	------	---------

0 Errors, 0 Warnings



Добавьте компоненты к системе

В появляющемся окне настройки каждого компонента нажмите Finish не изменяя настройки компонента - настройка компонентов будет осуществлена после их добавления к системе

- my_master
- my_slave
- my_slaveWS

При добавлении компонентов на закладке Messages будут появляться сообщения об ошибках. На данном этапе на них можно не обращать внимание.



Проверьте систему


- Убедитесь в том, что Ваша система выглядит так же, как представленная на рисунке
- Сохраните файл.

System: Lab3_1 Path: clk_0						
Use	Connections	Name	Description	Export	Clock	Base
<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/> clk_0	Clock Source			
		clk_in	Clock Input	clk	exported	
		clk_in_reset	Reset Input	reset		
		clk	Clock Output	Double-click to export	clk_0	
		clk_reset	Reset Output	Double-click to export		
<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/> my_master_0	my_master			
		clock	Clock Input	Double-click to export	unconnected	
		reset	Reset Input	Double-click to export	[clock]	
		m0	Avalon Memory Mapped Master	Double-click to export	[clock]	
<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/> my_slave_0	my_slave			
		clock	Clock Input	Double-click to export	unconnected	
		reset	Reset Input	Double-click to export	[clock]	
		s0	Avalon Memory Mapped Slave	Double-click to export	[clock]	
		conduit_end_0	Conduit	Double-click to export	[clock]	
<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/> my_slaveWS_0	my_slaveWS			
		clock	Clock Input	Double-click to export	unconnected	
		reset	Reset Input	Double-click to export	[clock]	
		s0	Avalon Memory Mapped Slave	Double-click to export	[clock]	
		conduit_end_0	Conduit	Double-click to export	[clock]	



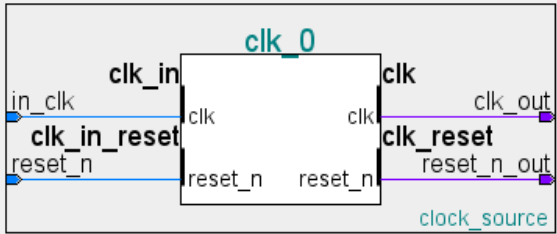
Настройка компонента **clk_0**

- Щелчком выберите **clk_0**
- Нажмите правую клавишу мыши и выберите команду Edit
- В появившемся окне задайте Reset synchronous edges = Deassert

 Clock Source
clock_source

Block Diagram

☒ Show signals



Parameters

Clock frequency: 50000000 Hz

☒ Clock frequency is known

Reset synchronous edges: Deassert ▾



Подключите тактовый сигнал

- На закладке System Contents щелчком выделите интерфейс **clk_0.clk** (интерфейс clk компонента clk_0)
- Выполните команду меню View=>Connections
- В появившемся окне закладки Connections выберите подключение ко всем тактовым входам
- Переключитесь на закладку System Contents
- Нажмите правую клавишу мыши
- Выберите команду Filter=>Clock and Reset Interfaces
- Убедитесь, что соединения выполнены -
Ваша система выглядит так же, как представленная на рисунке

System: Lab3_1 Path: clk_0.clk


Connected to: clk_0.clk

Connected	Connection	Clock Crossing
<input checked="" type="checkbox"/>	clk_0.clk/my_master_0.clock	
<input checked="" type="checkbox"/>	clk_0.clk/my_slaveWS_0.clock	
<input checked="" type="checkbox"/>	clk_0.clk/my_slave_0.clock	












Use	Con...	Name	Description	Export	Clock	Base
<input checked="" type="checkbox"/>		clk_0	Clock Source			
		clk_in	Clock Input	clk	<i>exported</i>	
		clk_in_reset	Reset Input	reset	[clk_in]	
		clk	Clock Output	<i>Double-click to export</i>	clk_0	
		clk_reset	Reset Output	<i>Double-click to export</i>	clk_0	
<input checked="" type="checkbox"/>		my_master_0	my_master			
		clock	Clock Input	<i>Double-click to export</i>	clk_0	
		reset	Reset Input	<i>Double-click to export</i>	[clock]	
<input checked="" type="checkbox"/>		my_slave_0	my_slave			
		clock	Clock Input	<i>Double-click to export</i>	clk_0	
		reset	Reset Input	<i>Double-click to export</i>	[clock]	
<input checked="" type="checkbox"/>		my_slaveWS_0	my_slaveWS			
		clock	Clock Input	<i>Double-click to export</i>	clk_0	
		reset	Reset Input	<i>Double-click to export</i>	[clock]	




Подключите сигнал Reset

- На закладке System Contents выполните команду меню System=>Create Global Reset Network
- Убедитесь, что соединения выполнены - Ваша система выглядит так же, как представленная на рисунке
- Сбросьте фильтрацию – нажмите на иконку  в нижней части окна System Contents
- Сохраните файл

System: Lab3_1 Path: clk_0.clk_reset

Use	Con...	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_in]	
		clk	Clock Output	<i>Double-click to export</i>	clk_0	
		clk_reset	Reset Output	<i>Double-click to export</i>	clk_0	
<input checked="" type="checkbox"/>		 my_master_0	my_master			
		clock	Clock Input	<i>Double-click to export</i>	clk_0	
		reset	Reset Input	<i>Double-click to export</i>	[clock]	
<input checked="" type="checkbox"/>		 my_slave_0	my_slave			
		clock	Clock Input	<i>Double-click to export</i>	clk_0	
		reset	Reset Input	<i>Double-click to export</i>	[clock]	
<input checked="" type="checkbox"/>		 my_slaveWS_0	my_slaveWS			
		clock	Clock Input	<i>Double-click to export</i>	clk_0	
		reset	Reset Input	<i>Double-click to export</i>	[clock]	

 **Current filter:** Clock and Reset Interfaces



Подключите Avalon-MM интерфейсы

- На закладке System Contents щелчком выделите интерфейс **my_master_0.m0**
- Нажмите правую клавишу мыши
- Выберите команду **Filter=> Avalon-MM Interfaces**
- В столбце Connections выполните подключения так, как показано на рисунке
- Сохраните файл

System: Lab3_1 Path: my_master_0.m0


Use	C...	Name	Description	Export	Clock	Base	End
<input checked="" type="checkbox"/>		<input type="checkbox"/> my_master_0	my_master		[clock]		
		<input type="checkbox"/> m0	Avalon Memor...	<i>Double-click to export</i>	clk_0		
<input checked="" type="checkbox"/>		<input type="checkbox"/> my_slave_0	my_slave		[clock]		
		<input type="checkbox"/> s0	Avalon Memor...	<i>Double-click to export</i>	clk_0	0x0000_0000	0x0000_0000
<input checked="" type="checkbox"/>		<input type="checkbox"/> my_slaveWS_0	my_slaveWS		[clock]		
		<input type="checkbox"/> s0	Avalon Memor...	<i>Double-click to export</i>	clk_0	0x0000_0000	0x0000_0003

Messages



Type	Path	Message
<input checked="" type="checkbox"/> 1 Error		
<input checked="" type="checkbox"/>	Lab3_1.my_master_0.m0	my_slaveWS_0.s0 (0x0..0x3) overlaps my_slave_0.s0 (0x0..0x0)



Назначьте базовые адреса ведомым Avalon-MM

- Компоненту `my_slave_0.s0` назначьте базовый адрес = 0
 - Дважды щелкните в поле Base адрес и введите 0
 - Зафиксируйте адрес – нажмите на символ 
- Выберите команду меню **System=> Assign Base Addresses**
- Убедитесь, что адреса назначены правильно - Ваша система выглядит так же, как представленная на рисунке

System: Lab3_1 Path: my_master_0.m0

Use	C...	Name	Description	Export	Clock	Base	End
<input checked="" type="checkbox"/>		<input type="checkbox"/> my_master_0	my_master		[clock]		
		<input type="checkbox"/> m0	Avalon Memor...	<i>Double-click to export</i>	clk_0		
<input checked="" type="checkbox"/>		<input type="checkbox"/> my_slave_0	my_slave		[clock]		
		<input type="checkbox"/> s0	Avalon Memor...	<i>Double-click to export</i>	clk_0	 0x0000_0000	0x0000_0000
<input checked="" type="checkbox"/>		<input type="checkbox"/> my_slaveWS_0	my_slaveWS		[clock]		
		<input type="checkbox"/> s0	Avalon Memor...	<i>Double-click to export</i>	clk_0	 0x0000_0004	0x0000_0007

- Сохраните файл
- Сбросьте фильтрацию – нажмите на иконку  в нижней части окна System Contents



Экспортируйте выводы

- На закладке System Contents щелчком выделите интерфейс `my_slave_0.conduit_end_0`
 - Дважды щелкните в поле Export и задайте имя `dout_a`
- На закладке System Contents щелчком выделите интерфейс `my_slaveWS_0.conduit_end_0`
 - Дважды щелкните в поле Export и задайте имя `dout_a`
- Сохраните файл

Use	Connections	Name	Description	Export	Clock	Base	End
<input checked="" type="checkbox"/>		my_slave_0	my_slave				
		clock	Clock Input	Double-click to export	clk_0		
		reset	Reset Input	Double-click to export	[clock]		
		s0	Avalon Memor...	Double-click to export	[clock]	0x0000_0000	0x0000_0000
		conduit_end_0	Conduit	dout_a	[clock]		
<input checked="" type="checkbox"/>		my_slaveWS_0	my_slaveWS				
		clock	Clock Input	Double-click to export	clk_0		
		reset	Reset Input	Double-click to export	[clock]		
		s0	Avalon Memor...	Double-click to export	[clock]	0x0000_0004	0x0000_0007
		conduit_end_0	Conduit	dout_b	[clock]		



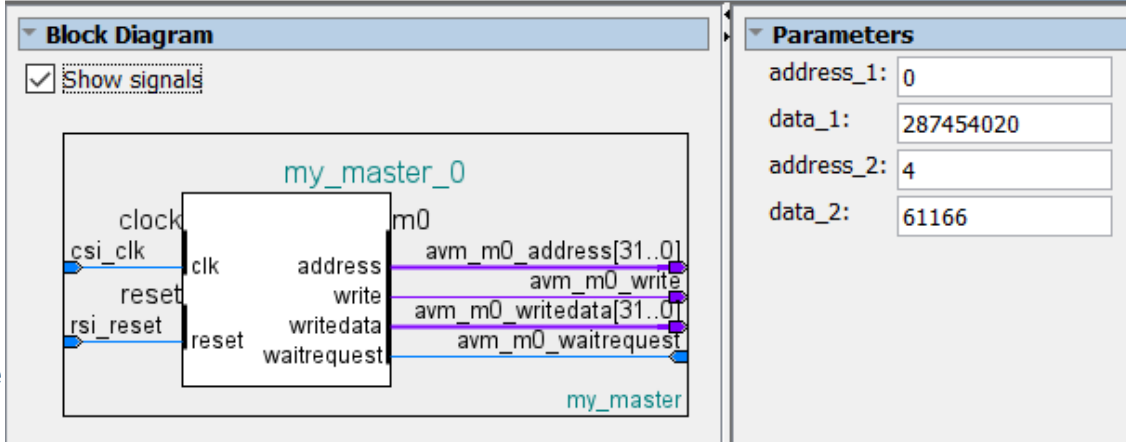
Настройка компонента my_master_0

- Щелчком выберите **my_master_0**
- Нажмите правую клавишу мыши и выберите команду **Edit**
- В появившемся окне задайте
 - **address_1:** = 0
 - **data_1:** = 287454020
 - Что соответствует 32'h11223344
 - **address_2:** = 4
 - **data_2:** = 61166
 - Что соответствует 32'h0000eeee
- Нажмите кнопку **Finish**
- Сохраните файл

my_master - my_master_0



my_master
my_master





Проверьте систему

- Убедитесь в том, что
 - Ваша система выглядит так же, как представленная на рисунке
 - Закладка сообщений (Messages) не содержит сообщений.

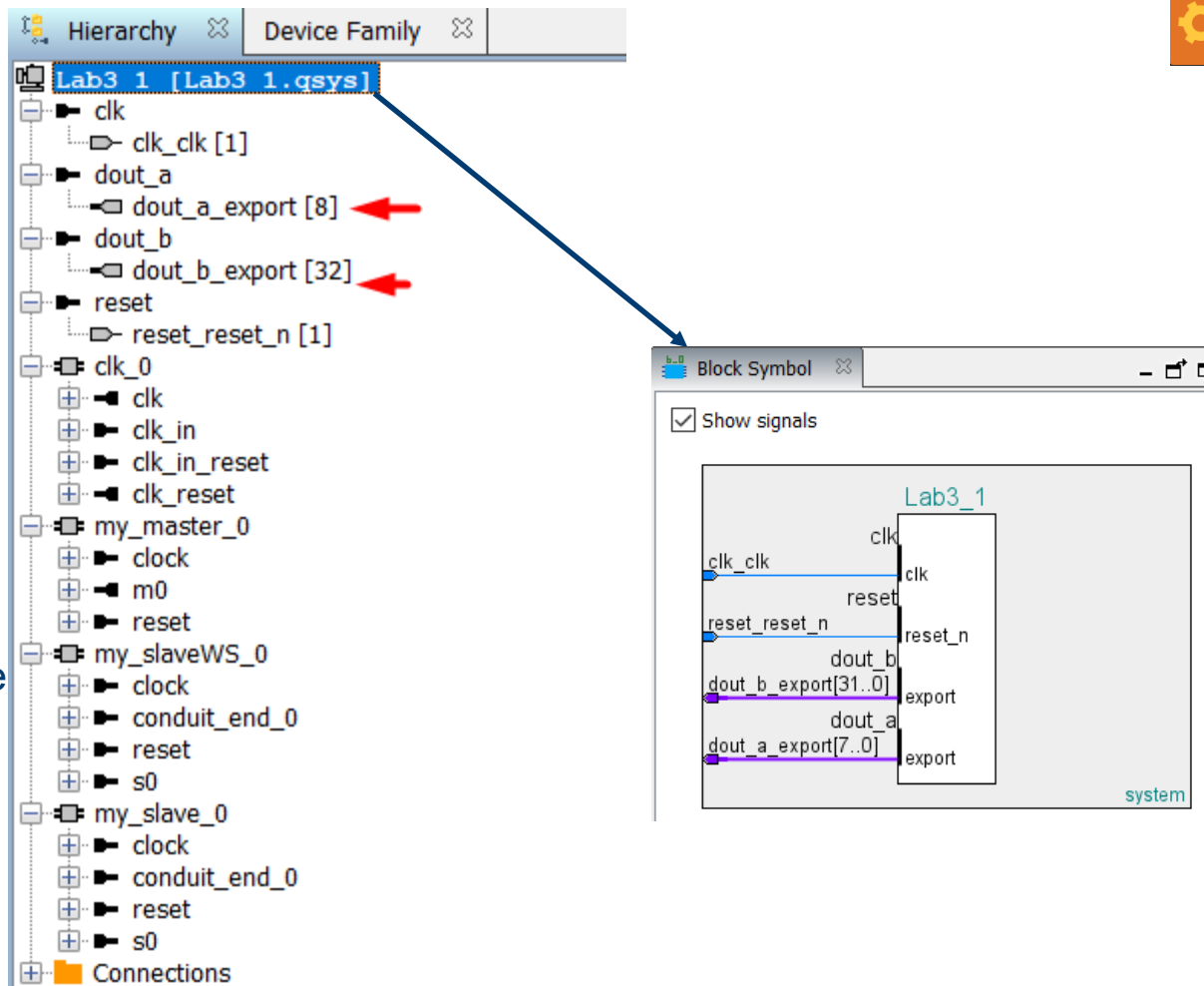
System: Lab3_1 Path: clk_0

Use	Connections	Name	Description	Export	Clock	Base	End
<input checked="" type="checkbox"/>		clk_0	Clock Source				
		clk_in	Clock Input	clk	exported		
		clk_in_reset	Reset Input	reset	[clk_in]		
		clk	Clock Output	Double-click to export	clk_0		
		clk_reset	Reset Output	Double-click to export	clk_0		
<input checked="" type="checkbox"/>		my_master_0	my_master				
		clock	Clock Input	Double-click to export	clk_0		
		reset	Reset Input	Double-click to export	[clock]		
		m0	Avalon Memor...	Double-click to export	[clock]		
<input checked="" type="checkbox"/>		my_slave_0	my_slave				
		clock	Clock Input	Double-click to export	clk_0		
		reset	Reset Input	Double-click to export	[clock]		
		s0	Avalon Memor...	Double-click to export	[clock]		
		conduit_end_0	Conduit	dout_a	[clock]	0x0000_0000	0x0000_0000
<input checked="" type="checkbox"/>		my_slaveWS_0	my_slaveWS				
		clock	Clock Input	Double-click to export	clk_0		
		reset	Reset Input	Double-click to export	[clock]		
		s0	Avalon Memor...	Double-click to export	[clock]		
		conduit_end_0	Conduit	dout_b	[clock]	0x0000_0004	0x0000_0007




Анализ системы

- Выполните команду: меню View=>Hierarchy
- В окне закладки Hierarchy выделите систему Lab3_sys [Lab3_sys.qsys]
- Выполните команду: меню View=>Block Symbol
- Убедитесь, что Ваш символ системы соответствует представленному на рисунке




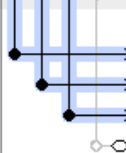




Анализ системы

- Выполните команду: меню View => Clock domains Beta
- Выберите режим отображения Clocks 
Clocks **Resets**
- Убедитесь в том, что в столбце Connections нет красных точек => нет проблем подключения

System: Lab3_1

Use	Connections	Name	Description	Export	Clock
<input checked="" type="checkbox"/>		<div>clk_0<ul style="list-style-type: none">clk_inclk_in_resetclkclk_reset</div>	<div>Clock Source<ul style="list-style-type: none">Clock InputReset InputClock OutputReset Output</div>	<div>clk reset <i>Double-click to export</i> <i>Double-click to export</i></div>	<div><i>exported</i> [clk_in] clk_0 clk_0</div>
<input checked="" type="checkbox"/>		<div>my_master_0<ul style="list-style-type: none">clockresetm0</div>	<div>my_master<ul style="list-style-type: none">Clock InputReset InputAvalon Memor...</div>	<div><i>Double-click to export</i> <i>Double-click to export</i> <i>Double-click to export</i></div>	<div>clk_0 [clock] [clock]</div>
<input checked="" type="checkbox"/>		<div>my_slave_0<ul style="list-style-type: none">clockresets0conduit_end_0</div>	<div>my_slave<ul style="list-style-type: none">Clock InputReset InputAvalon Memor...Conduit</div>	<div><i>Double-click to export</i> <i>Double-click to export</i> <i>Double-click to export</i></div>	<div>clk_0 [clock] [clock] [clock]</div>
<input checked="" type="checkbox"/>		<div>my_slaveWS_0<ul style="list-style-type: none">clockresets0conduit_end_0</div>	<div>my_slaveWS<ul style="list-style-type: none">Clock InputReset InputAvalon Memor...Conduit</div>	<div><i>Double-click to export</i> <i>Double-click to export</i> <i>Double-click to export</i></div>	<div>clk_0 [clock] [clock] [clock]</div>



Анализ системы

- Выберите режим отображения Reset
- Убедитесь в том, что в столбце Connections нет красных точек => нет проблем подключения



Clocks **Resets**

System: Lab3_1					
Use	Connections	Name	Description	Export	Clock
<input checked="" type="checkbox"/>		<input type="checkbox"/> clk_0	Clock Source		
		clk_in	Clock Input	clk	<i>exported</i>
		clk_in_reset	Reset Input	reset	[clk_in]
		clk	Clock Output	<i>Double-click to export</i>	clk_0
		clk_reset	Reset Output	<i>Double-click to export</i>	clk_0
<input checked="" type="checkbox"/>		<input type="checkbox"/> my_master_0	my_master		
		clock	Clock Input	<i>Double-click to export</i>	clk_0
		reset	Reset Input	<i>Double-click to export</i>	[clock]
<input checked="" type="checkbox"/>		<input type="checkbox"/> my_slave_0	my_slave		
		clock	Clock Input	<i>Double-click to export</i>	clk_0
		reset	Reset Input	<i>Double-click to export</i>	[clock]
		s0	Avalon Memor...	<i>Double-click to export</i>	[clock]
		conduit_end_0	Conduit	dout_a	[clock]
<input checked="" type="checkbox"/>		<input type="checkbox"/> my_slaveWS_0	my_slaveWS		
		clock	Clock Input	<i>Double-click to export</i>	clk_0
		reset	Reset Input	<i>Double-click to export</i>	[clock]
		s0	Avalon Memor...	<i>Double-click to export</i>	[clock]
		conduit_end_0	Conduit	dout_b	[clock]



Анализ системы

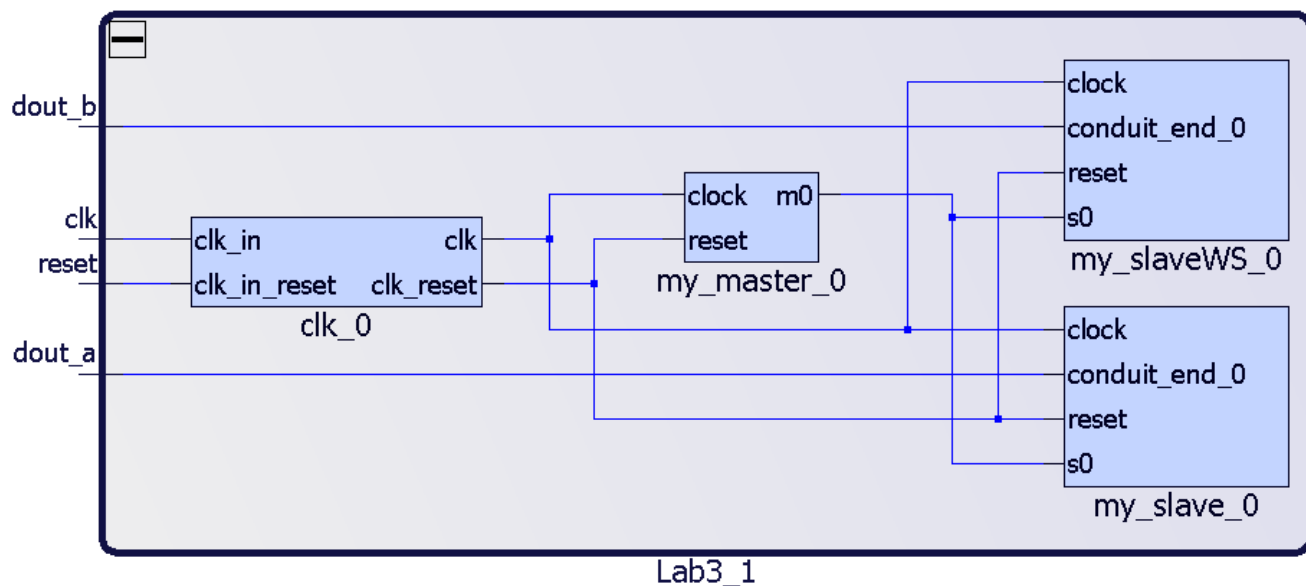
- Выполните команду: меню System => Show System with Platform Designer Interconnect сравните созданную Вами систему и систему с модулями добавленными PD:
 - Убедитесь в том, что PD добавил только модуль mm_interconnect

System: Lab3_1							
Use	Connections	Name	Description	Export	Clock	Base	End
<input checked="" type="checkbox"/>		mm_interconnect_0	MM Interconnect				
		clk_0_clk	Clock Input	<i>Double-click to export</i>	clk_0		
		my_master_0_reset_reset...	Reset Input	<i>Double-click to export</i>	[clk_0_clk]		
		my_master_0_m0	Avalon Memor...	<i>Double-click to export</i>	[clk_0_clk]	0x0000_0000	0xffff_ffff
		my_slave_0_s0	Avalon Memor...	<i>Double-click to export</i>	[clk_0_clk]		
		my_slaveWS_0_s0	Avalon Memor...	<i>Double-click to export</i>	[clk_0_clk]		
<input checked="" type="checkbox"/>		clk_0	Clock Source				
		clk_in	Clock Input	clk	exported		
		clk_in_reset	Reset Input	reset	[clk_in]		
		clk	Clock Output	<i>Double-click to export</i>	clk_0		
		clk_reset	Reset Output	<i>Double-click to export</i>	clk_0		
<input checked="" type="checkbox"/>		my_master_0	my_master				
		clock	Clock Input	<i>Double-click to export</i>	clk_0		
		reset	Reset Input	<i>Double-click to export</i>	[clock]		
		m0	Avalon Memor...	<i>Double-click to export</i>	[clock]		
<input checked="" type="checkbox"/>		my_slave_0	my_slave				
		clock	Clock Input	<i>Double-click to export</i>	clk_0		
		reset	Reset Input	<i>Double-click to export</i>	[clock]		
		s0	Avalon Memor...	<i>Double-click to export</i>	[clock]	0x0000	0x0000
		conduit_end_0	Conduit	dout_a	[clock]		
<input checked="" type="checkbox"/>		my_slaveWS_0	my_slaveWS				
		clock	Clock Input	<i>Double-click to export</i>	clk_0		
		reset	Reset Input	<i>Double-click to export</i>	[clock]		
		s0	Avalon Memor...	<i>Double-click to export</i>	[clock]	0x0000	0x0003
		conduit_end_0	Conduit	dout_b	[clock]		



Анализ системы

- Выполните команду: меню **View=>Schematic**
- Убедитесь в том, что система синхронизации Вашей системы выглядит так же, как представленная на рисунке

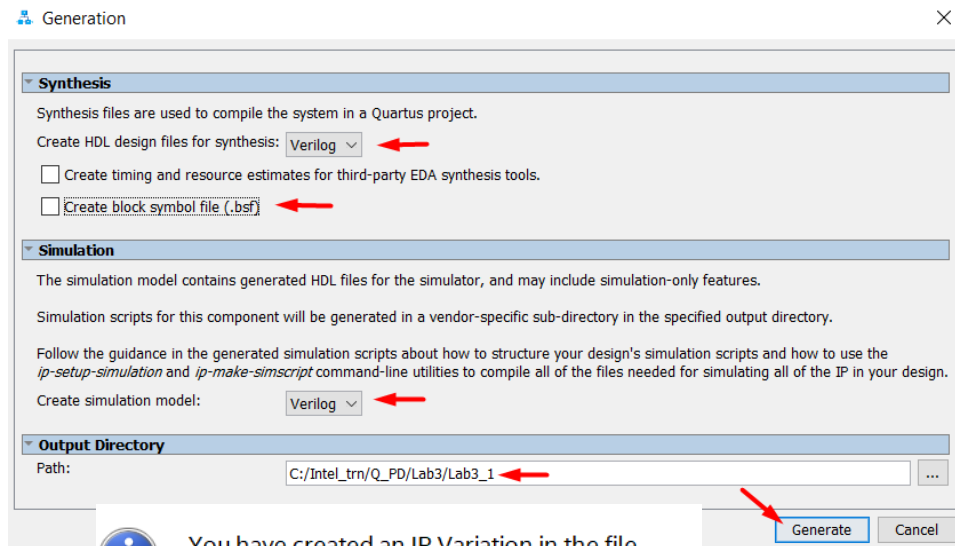




Генерация системы

- В окне PD нажмите кнопку Generate HDL... (правый нижний угол окна)
- Установите режимы так, как показано на рисунке
- Нажмите кнопку Generate
- По окончании процедуры появится сообщение
 - Нажмите кнопку Close
- В окне PD нажмите кнопку Finish (правый нижний угол окна)
- Появится напоминание о необходимости подключить файлы к проекту пакета QP
- Нажмите кнопку ОК.

Generate: completed successfully.



You have created an IP Variation in the file
C:/Intel_trn/Q_PD/Lab3/Lab3_1.qsys.

To add this IP to your Quartus project, you must manually add the .qip and .sip files after generating the IP core.

The .qip will be located in
<generation_directory>/synthesis/Lab3_1.qip

The .sip will be located in
<generation_directory>/simulation/Lab3_1.sip



Подключите файлы к проекту в QP

- В QP
 - Выполните **Project => Add\Remove Files from project**
 - Lab3_1.qip
 - Lab3_1.sip
 - Lab3_top.sv

Category:

General

Files

Libraries

▼ IP Settings

IP Catalog Search Locations

Design Templates

▼ Operating Settings and Conditions

Voltage

Temperature

▼ Compilation Process Settings

Incremental Compilation

Files

Select the design files you want to include in the project. Click Add All to add all design f

File name:



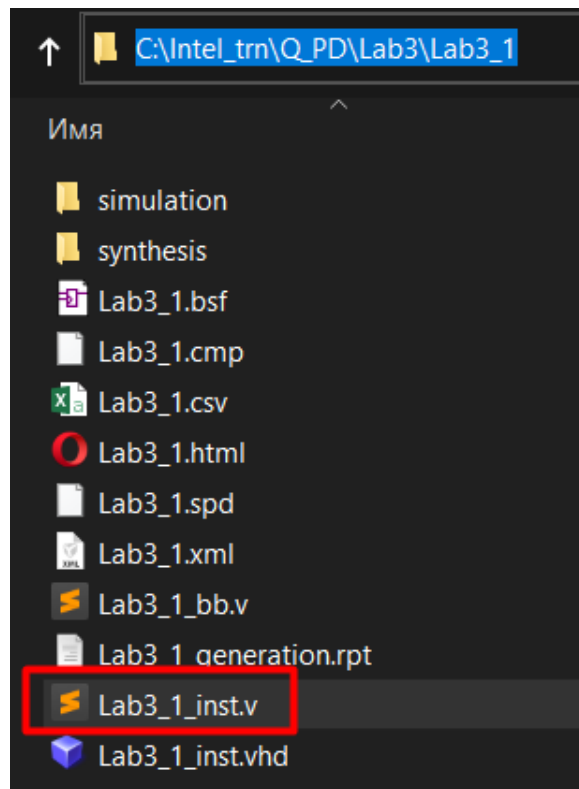
File Name	Type	Library	Design Ent
Lab3_top.sv	SystemVerilog HDL File		<None>
Lab3_1/simulation/Lab3_1.sip	Quartus Prime SIP File		<None>
Lab3_1/synthesis/Lab3_1.qip	IP Variation File (.qip)		<None>



Файл Lab3_top.sv

- Создан с использование файла Lab3_1_inst.v

```
1  `timescale 1 ns / 1 ns
2  module Lab3_top (
3      input bit clk,
4      input bit reset,
5      output bit [7:0] ledA,
6      output bit [31:0] ledB
7  );
8  Lab3_1 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
```

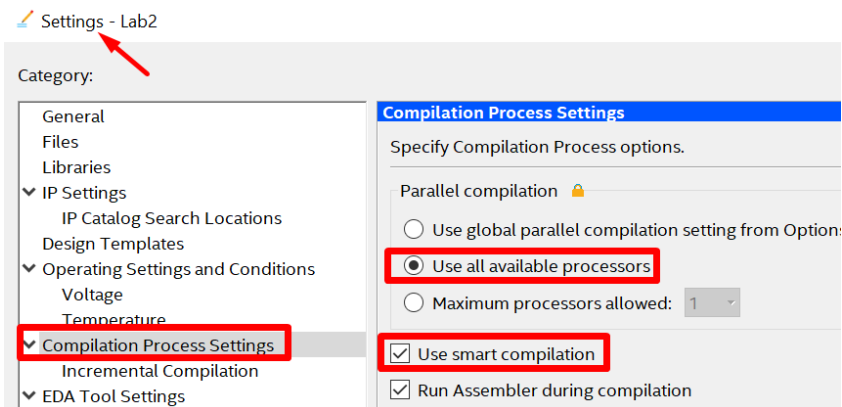
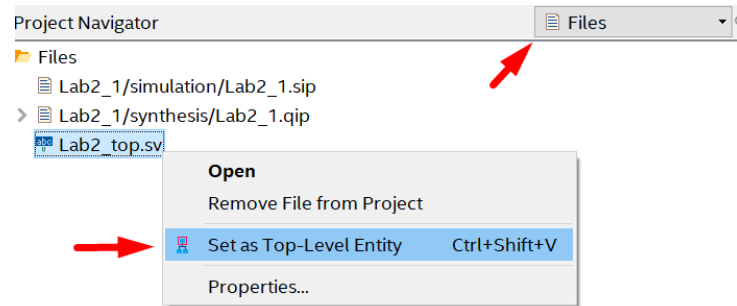
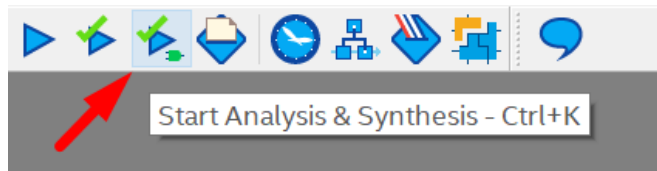




Анализ и синтез в QP

■ В QP

- Файл Lab2_top.sv объявите файлом верхнего уровня
- Выполните назначения, показанные на рисунке. Команда: **меню Assignment=>Settings**
- Выполните команду **Start Analysis and Synthesis**

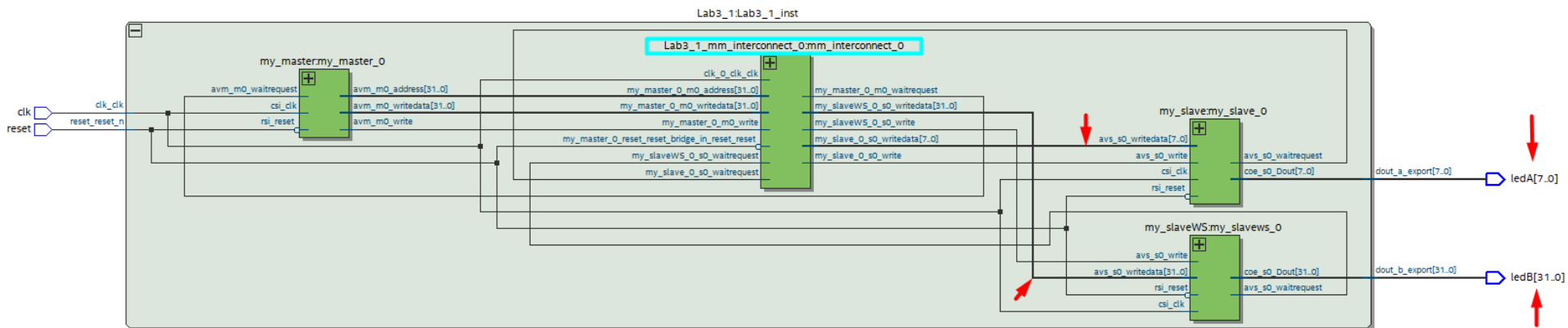


*Убедитесь в том, что
компиляция без ошибок*



Анализ RTL Viewer

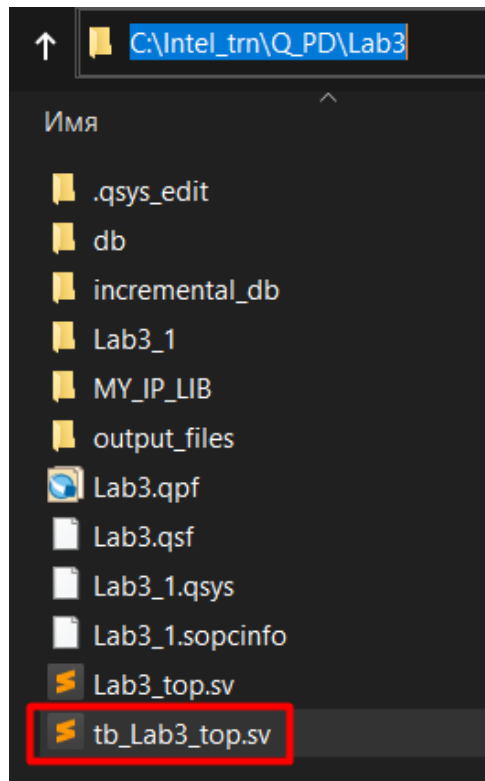
- Выполните: меню Tools=>Netlist Viewers => RTL viewer
- Убедитесь в том, что Ваша схема похожа на схему, приведенную на рисунке





Файл tb_Lab3_top.sv

- Тест для проверки системы

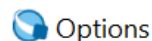


```
1  `timescale 1 ns / 1 ns
2  module tb_Lab3_top ();
3      bit clk;
4      bit reset;
5      bit[7:0]    ledA;
6      bit[31:0]   ledB;
7
8      always #50 clk = ~ clk;
9
10     initial
11     begin
12         clk      = 1'b0;
13         reset     = 1'b0;
14         #200;
15         reset     = 1'b1;
16         #1000;
17         $stop;
18     end
19
20     Lab3_top Lab3_top_inst (.*) ;
21
22 endmodule
```



Настройка QP для NativeLink

- Убедитесь, что правильно задана ссылка на пакет ModelSim
 - Выполните команду Tools=>Options



Category:

- ▼ General
 - EDA Tool Options**
 - Fonts
 - Headers & Footers Settings
- ▼ Internet Connectivity
 - Notifications
 - Libraries
- ▼ IP Settings
 - IP Catalog Search Locations
 - Design Templates
 - License Setup
 - Preferred Text Editor
 - Processing
 - Tooltip Settings
- ▼ Messages

EDA Tool Options

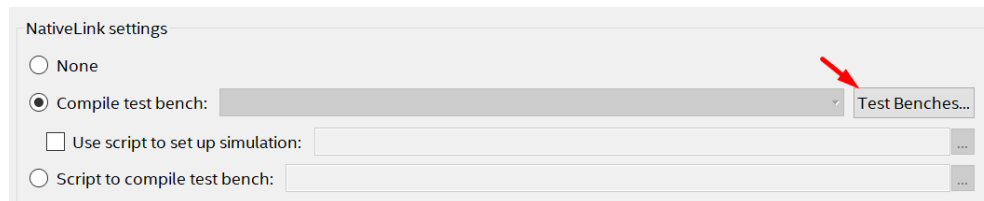
Specify the directory that contains the tool executable for each third-party ED.

EDA Tool	Directory Containing Tool Executable
Precision ...	
Synplify	
Synplify ...	
Active-HDL	
Riviera-P...	
ModelSim	
QuestaSim	
ModelSi...	C:\intelFPGA_lite\20.1\modelsim_ase\win32aloem



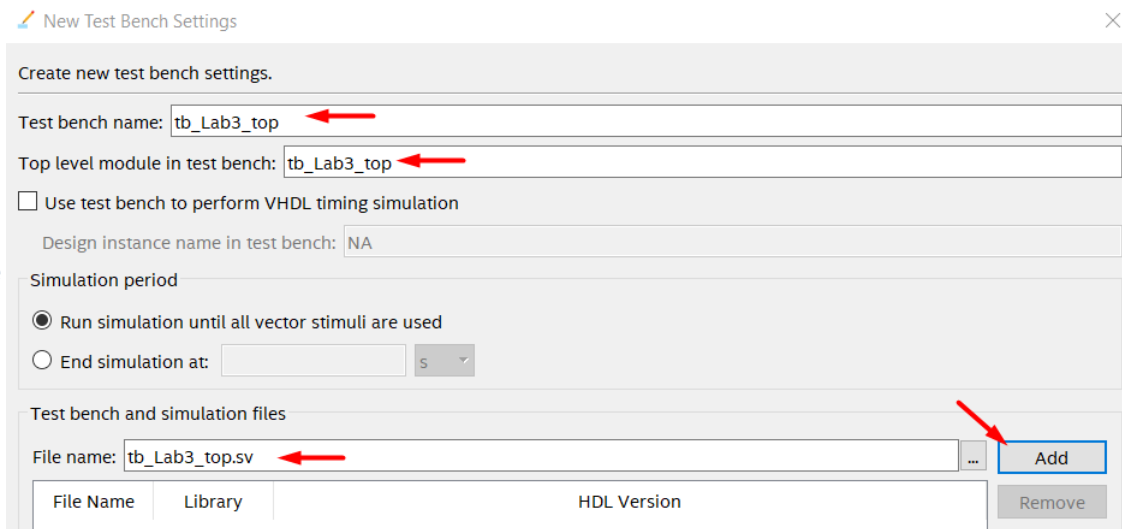
Настройка QP для NativeLink

- Выполните команду : **меню Assignment=>Settings=>Simulation =>NativeLink settings=>кнопка Test Benches**



- Нажмите кнопку **New**

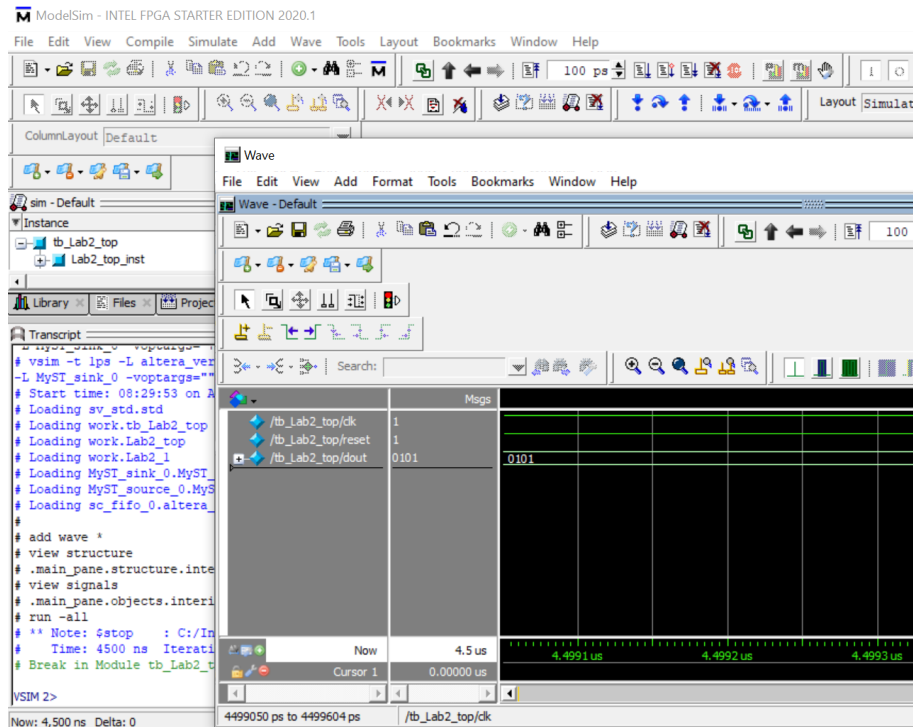
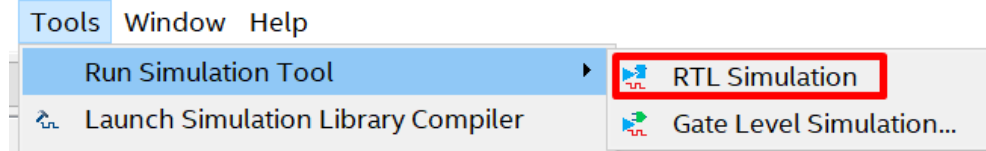
- Выполните назначения, показанные на рисунке.





Запуск моделирования с NativeLink

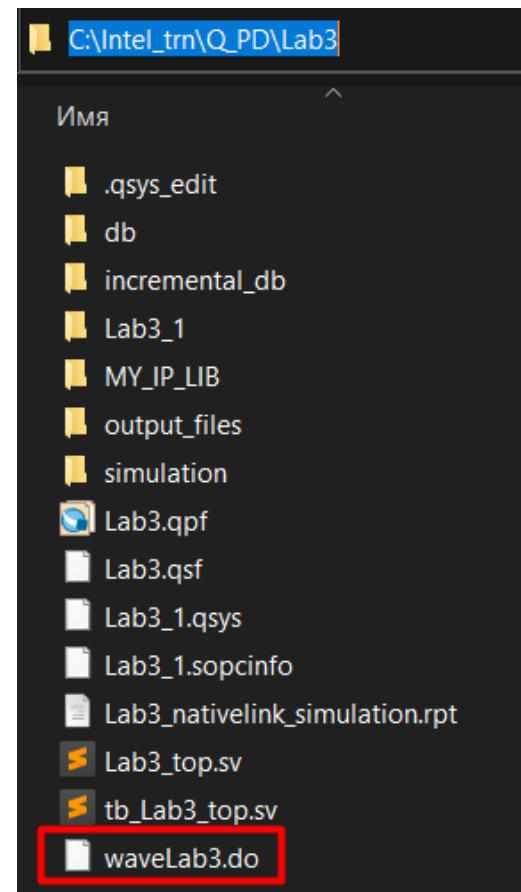
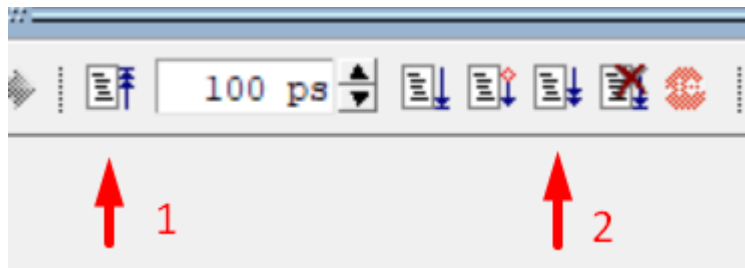
- Выполните команду : меню **Tools=>Run Simulation Tool=>RTL Simulation**
- Откроется окно (окна) пакета ModelSim





Загрузка формата временной диаграммы

- В окне Wave пакета ModelSim выполните команду: File=>Load и выберите файл waveLab3.do
- В окне Wave пакета ModelSim нажмите кнопку Restart а затем Run -All





-





Лабораторная 3
ЗАВЕРШЕНА!