**Подключение периферийных устройств и расширение системы**

Если рассмотреть популярные варианты компоновки систем с аппаратными или программными процессорными ядрами, то можно увидеть примерно следующее.

Популярным решением во многих IDE различных производителей FPGA является шина AXI – так, например, в Vivado достаточно много IP-ядер завязанных на AXI.

Многие технические решения, особенно по замещению процессорных ядер уже в микроконтроллерных решениях является шина AHB – конечно, не в рамках FPGA, но в микроконтроллерах китайского производства часто можно встретить микроконтроллеры-дубли, в которых ARM ядро замещено ядром RISC-V или вообще есть возможность выбора при включении, какое из ядер будет активно.

Рассмотренный ранее софт-процессор SCR1 также для взаимодействия с внешними устройствами использует в зависимости от конфигурации или шину AXI, или AHB/

В Open-Source проектах и их экосистеме популярны IP-ядра для шины Wishbone.

Ниже попробуем кратко рассмотреть особенности каждой из упомянутых шин.

**AXI**

AXI (Advanced eXtensible Interface) - продвинутый это протокол интерфейса, определенный ARM как часть стандарта AMBA (Advanced Microcontroller Bus Architecture) – рис.1.

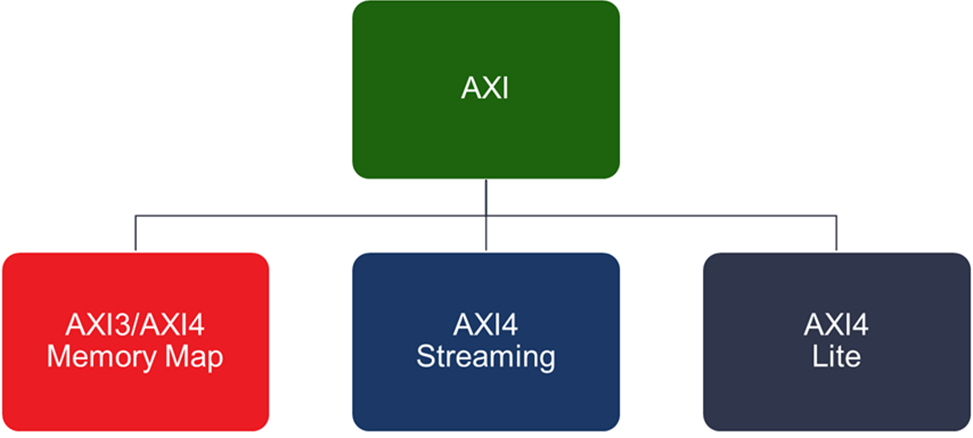


Рис. 1 Варианты AXI.

Протокол AXI4 был разработан для приложений с высокой пропускной способностью и низкой задержкой. Он предназначен для обеспечения связи между ведущим и ведомыми устройствами. Главным устройством обычно является DMA или центральный процессор, а подчиненными - контроллеры DRAM или контроллеры других специальных протоколов: UART, SPI и другие. Иногда один компонент может реализовывать несколько экземпляров этого протокола. Обычно префикс используется для различения нескольких интерфейсов AXI4.

Существует 3 типа AXI4-интерфейсов (AMBA 4.0):

- AXI4 (Полный AXI4): для высокоскоростных устройств, как правило отображаемых в память процессорной системы (memory-mapped);

- AXI4-Lite: для простой передачи данных с использованием памяти с низкой пропускной способностью (например, в регистры управления и состояния и из них);

- AXI4-Stream: для высокоскоростной потоковой передачи данных.

**Каналы AXI**

Протокол AXI определяет 5 канало, два используются для транзакций чтения;

- адреса чтения;

- данных.

Три канала используются для транзакций:

- записи;

- адреса записи, данных;

- ответа на запись.

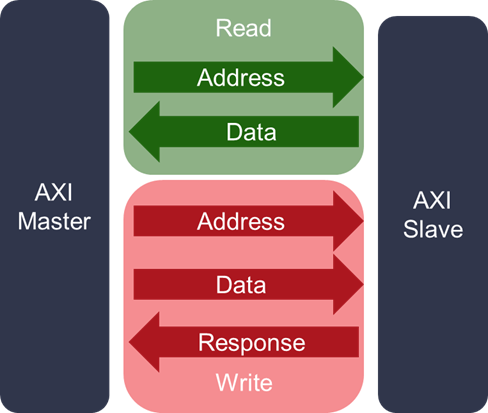


Рис. 2 Каналы AXI.

Канал - это независимая совокупность сигналов AXI, связанных с сигналами VALID и READY.

Фрагмент данных, передаваемый по одному каналу, называется передачей. Передача происходит, когда как VALID, так и READY сигнал находятся на высоком уровне, при фронте тактового сигнала Например, на рисунке 3 ниже передача выполняется на такте T3.

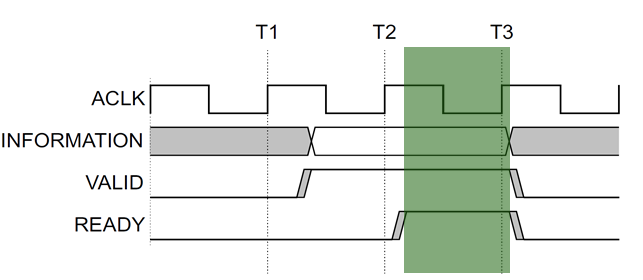


Рис. 3 Пример транзакции.

Транзакции чтения AXI требуют многократного обмена сигналами по двум каналам чтения. Сначала на канал считывания адреса передается от ведущего устройства к ведомому для установки адреса и некоторых управляющих сигналов. Затем данные для этого адреса передаются от ведомого устройства к ведущему устройству по каналу считывания данных. Обратите внимание, на рис. 4 при чтении одного адреса может быть передано несколько данных. Этот тип транзакции называется пакетной (brust).

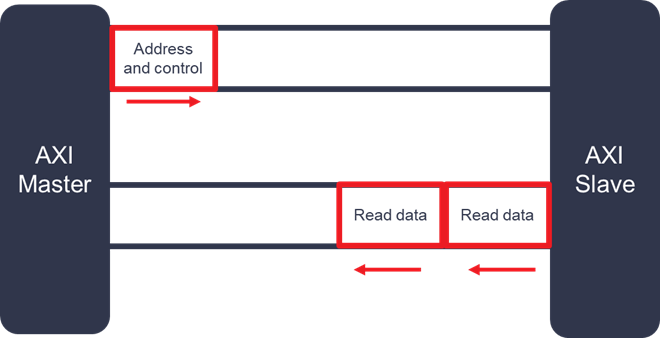


Рис. 4 Пример пакетного чтения данных.

Транзакции записи AXI также требуют многократной передачи данных по трем каналам считывания. Сначала канал записи адреса передается ведущим на ведомое устройство для установки адреса и некоторых управляющих сигналов. Затем данные для этого адреса передаются ведущим на ведомое устройство по каналу записи данных. Наконец, ответ на запись отправляется от ведомого устройства к ведущему по каналу ответа на запись, чтобы указать, была ли передача успешной.

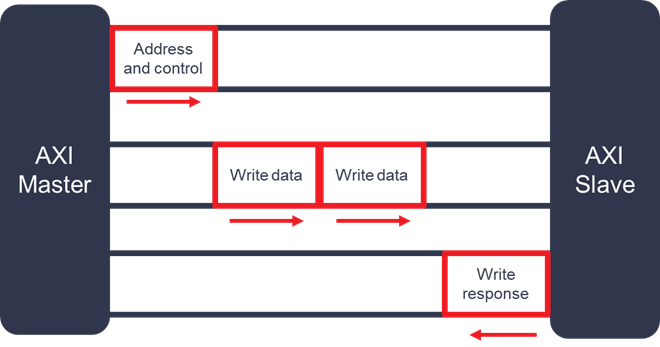


Рис. 5 Транзакции по AXI при операциях записи.

Возможные значения ответа в канале ответа на запись:

OKAY (0b00): Обычный доступ завершен успешно. Указывает, что обычный доступ был выполнен успешно.

EX\_OKAY (0b01): Исключительный доступ в порядке.

SLVERR (0b10): Ошибка ведомого устройства. Соединение с подчиненным устройством было достигнуто успешно, но подчиненное устройство хочет вернуть исходному ведущему устройству сообщение об ошибке (например, недопустимые считанные данные).

DECERR (0b11): Ошибка декодирования. Генерируется, как правило, компонентом арбитром шины, чтобы указать, что по адресу транзакции нет подчиненного устройства.

Примечание: Транзакции чтения также имеют значение ответа, но этот ответ передается в канале как часть ответа на чтение.

В спецификации AXI4 задокументированы некоторые требования. Ключевые требования следующие:

- выставлен сигнал VALID (AxVALID/ VALID), он должен оставаться высоким до положительного фронта тактового сигнала, после выставления ведомым сигналов AxREADY/xREADY;

- сигнал READY интерфейса AXI, передающей стороны, не должен зависеть от сигнала READY интерфейса AXI, принимающей стороны. Однако состояние сигнала READY может зависеть от сигнала VALID;

- ответ на запись всегда должен следовать за последней передачей записи в транзакции записи, частью которой он является;

- данные для чтения всегда должны следовать за адресом, к которому относятся данные;

- ведомый должен дождаться пока оба сигнала как ARVALID, так и ARREADY будут выставлены, прежде чем он выставит RVALID, чтобы указать, что данные действительны.

Каналы являются носителями транзакций по каналам. AXI4 реализует 5 каналов:

- Адрес для чтения - AR - канал от ведущего к подчиненному;

- Данные для чтения: - R - канал от ведомого к ведущему;

- Адрес для записи - AW - канал от ведущего к подчиненному;

- Данные для записи – W - канал от ведущего к подчиненному;

- Ответ на запись – B - канал от ведомого устройства к ведущему.

Каждый канал соответствует правилам канального протокола. Возможны варианты только для чтения и записи. Это достигается путем реализации только каналов, необходимых конкретному устройству. Спецификация AXI также определяет протокол AXI4-Lite, который устанавливает более строгие правила для транзакций, генерируемых ведущим устройством. К ведущему устройству AXI4-Lite может быть подключено любое подчиненное устройство AXI4, однако к подчиненным устройствам AXI4-Lite могут быть подключены только ведущие устройства AXI4, следующие правилам AXI4-Lite.

**Правила протокола**

Каждый канал имеет сигнал готовности/подтверждения (VALID/READY) и сигналы полезной нагрузки.

Каналы имеют свое направление, которое соответствует направлению сигнала VALID. Приемная сторона оперирует сигналом готовности READY, которое противоположно сигналу VALID. Для каждого канала требуется, чтобы было подтверждено значение VALID, сигнализирующее о начале транзакции по каналу. Принимающая сторона подтверждает сигнал READY, чтобы сообщить ведущему каналу, что транзакция по каналу принята и можно запускать следующую транзакцию по каналу. Транзакция по одному каналу определяется как выставление данных, стробируемых сигналом VALID, подтверждение приема определяется по сигналу READY, выставленному приемной стороной – на этом цикл обмена по шине считается завершённым. Сигналы полезной нагрузки канала (данных) не должны изменяться, пока транзакция активна, и сигналVALID не должен зависеть от значения сигнала READY. VALID не может быть отменен до тех пор, пока не будет подтвержден READY.

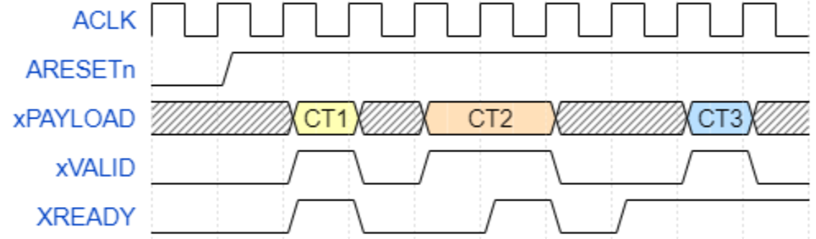


Рис.6 Пример трех циклов транзакций.

Преимущества разделения групп сигналов на каналы заключаются в упрощении пересечения доменов синхронизации, питания и сброса, а также в разделении регистров ("разделении периодов"). Это также дополнительно проясняет соотношение сигналов, поскольку разные каналы имеют разные допустимые сигналы и префиксы, трудно случайно неправильно понять, какие сигналы допустимы на каком этапе.

**Сигналы AXI4**

AXI4 требует, чтобы интерфейс реализовывал ACLK, который передает тактовый сигнал, и требовал, чтобы сигналы в каждом канале были действительны на восходящем фронте этого тактового сигнала. Также требуется реализовать ARESETn, который при низком уровне сигнала сбрасывает сигналы. Пока выполняется сброс, и через один цикл после сброса требуется понизить уровень сигналов допустимости/готовности всех каналов.

**Таблица 1**

**Сигналы каналов AW/AR от ведущего к ведомому**

|  |  |  |
| --- | --- | --- |
| Канал AW, название сигнала | Канал AR, название сигнала | Описание |
| AWVALID | ARVALID | Сигналы стробирования/подтверждения приема |
| AWREADY | AWVALID |
| AWID | ARID | Идентификатор несущей |
| AWADDR | ARADDR | Генерация адреса транзакции и сигналов, связанных с ним и параметрами транзакции. |
| AWLEN | ARLEN |
| AWSIZE | ARSIZE |
| AWBURST | ARBURST |
| AWPROT | ARPROT |
| AWLOCK | ARLOCK | Сигналы атомарных операций |
| AWCACHE | ARCACHE | Сигналы кеширования. |

Таблица 2

**Сигналы канала записи (W) – от ведущего к ведомому**

|  |  |
| --- | --- |
| Сигнал | Описание |
| WVALID | Стробирующие/подтверждающие сигналы |
| WREADY |
| WDATA | Запись данных |
| WSTRB | Строб записи (активный уровень - высокий) |
| WLAST | Signals last channel transaction for this transaction. |

Таблица 3

**Сигналы канала B – от ведомого к ведущему**

|  |  |
| --- | --- |
| Сигнал | Описание |
| BVALID | Стробирующие/подтверждающие сигналы |
| BREADY |
| BID | Описание несущей |
| BRESP | Ответ |

Таблица 4

**Сигналы канала R - от ведомого к ведущему**

|  |  |
| --- | --- |
| Сигнал | Описание |
| RVALID | Стробирующие/подтверждающие сигналы |
| RREADY |
| RID | Идентификатор |
| RDATA | Данные ответа |
| RRESP | Строб ответа |
| RLAST | Сигнал последней транзакции по каналу |

**Запросы на чтение**

Запросы на чтение AXI4 поступают по каналам AR и R. Канал AR используется ведущим устройством для отправки запросов на чтение подчиненному устройству. Ведущий сначала устанавливает сигналы канала AR, такие как RADDR, ARSIZE, ARBURST ARLEN и другие сигналы.

Затем выставляется сигнал ARVALID, сигнализирующий о запуске новой транзакции. Ведомое устройство подтверждает, что транзакция готова, сообщая ведущему устройству, что транзакция принята, и ответ будет возвращен по каналу R. Подчиненный после завершения цикла принимает транзакцию по каналу AR и устанавливает сигналы по каналу R, такие как RRESP, RDATA и RID (рис.2).

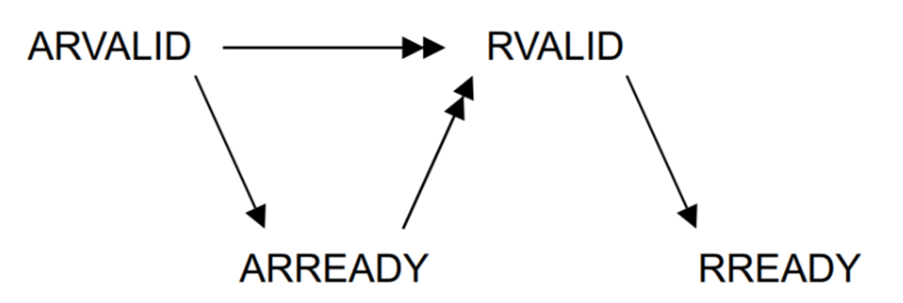


Рис. 7 Последовательность состояний при запросе на чтение.

За одной транзакцией по каналу AR может последовать несколько ответов. Каждая полезная нагрузка по каналу R называется "beat". Несколько beat с одним последним beat, подтверждающим последний сигнал, называются пакетом. Один запрос AR с одним пакетом по каналу R называется транзакцией чтения AXI.

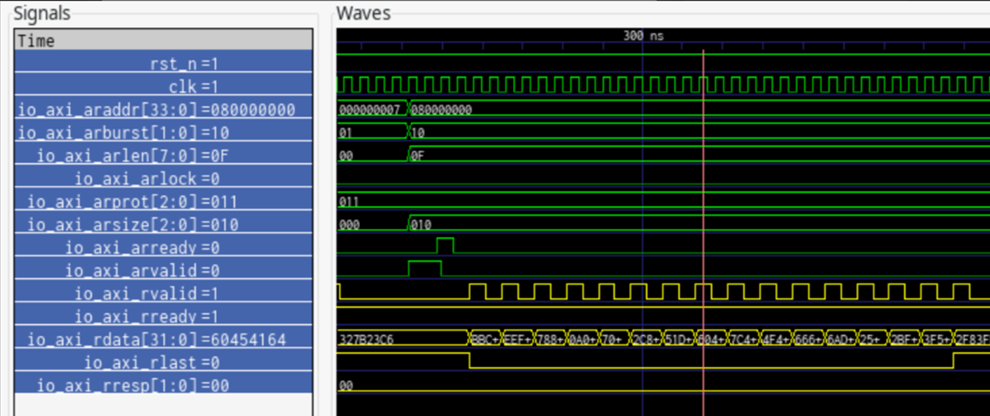


Рис. 8 Временные диаграммы при транзакции на AXI.

К сигналам R относятся следующие: R RESP, RDATA, RLAST, RID и, сигналы подтверждения связи. ARLEN содержит количество beat минус один. Ведомый использует эту информацию для генерации тактов данных ответа, соответствующих ARLEN + 1. Каждый beat RID должен содержать то же значение, что и ARID при запуске транзакции. Для компенсации задержек соединения и slave может быть запущено несколько транзакций. Подчиненные устройства должны отвечать в том же порядке, в каком был запрошен запрос, только если эти транзакции имеют одинаковые идентификаторы.

Примеры транзакций. В сигналах RDATA Dy показано, что для транзакции были возвращены данные с адреса y.

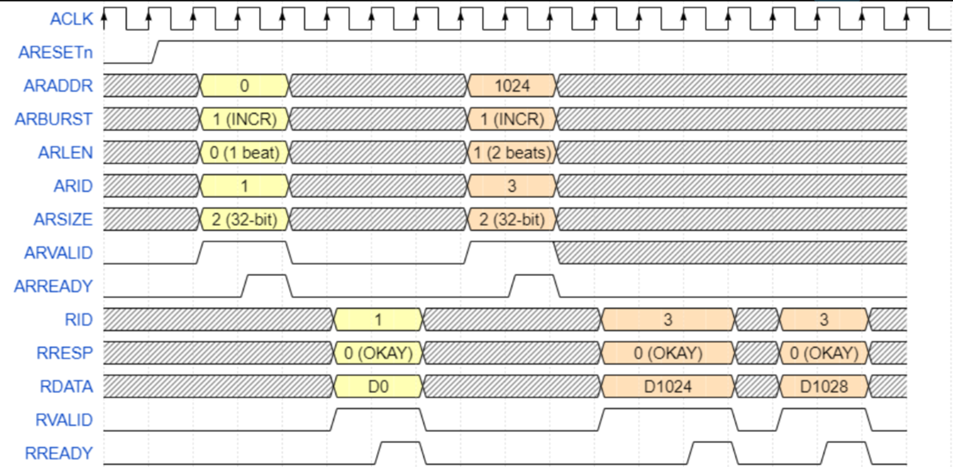


Рис. 9 Светло-желтым цветом – первая транзакция, представляет собой пакет длиной в один такт. Вторая - пакет длиной в два такта.

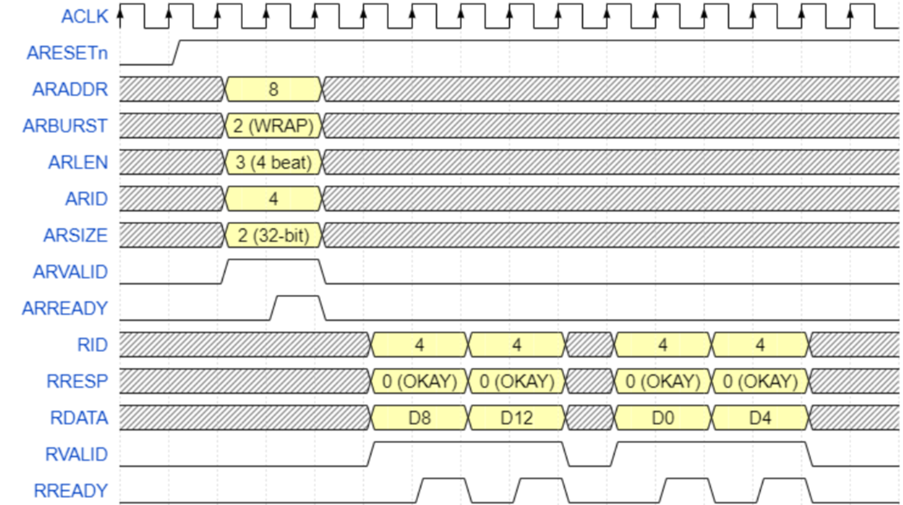


Рис. 10 Временная диаграмма сигналов в 4-тактной транзакции (тип beat).

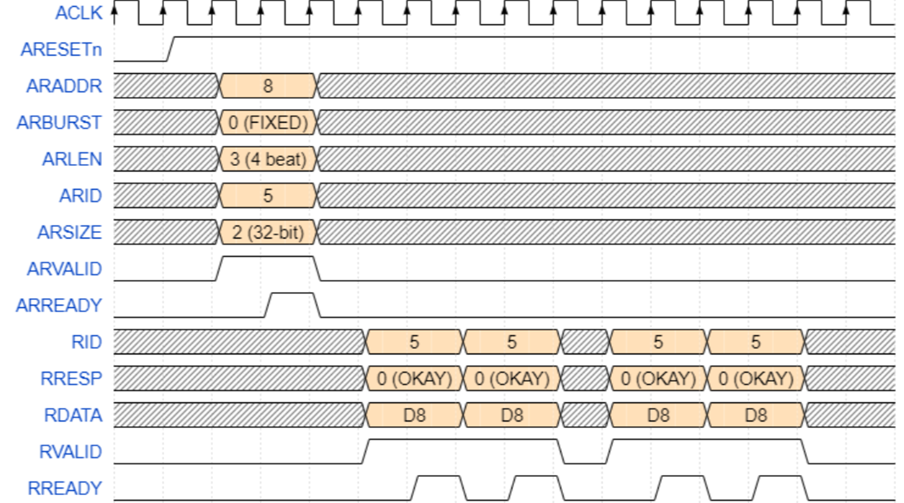


Рис. 11 Временные диаграммы 4-тактной транзакции (тип Fixed).

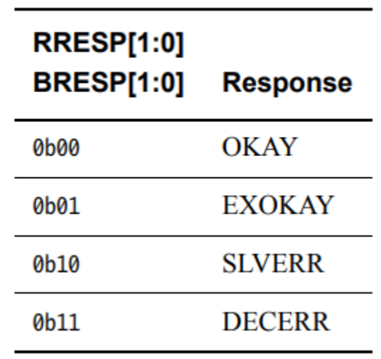


Рис.12 Значения сигналов RRESP/BRESP и их семантика.

**Транзакции записи**

Транзакции записи запускаются по запросу на канале AW. Ведущий устанавливает сигналы AW, такие как AWADDR, AWSIZE, ALLEN, AWBURST и т.д. Затем ведущий генерирует AWVALID. В то же время ведущий также устанавливает данные для W-канала, такие как WSTRB, DATA, WLAST. Ведущий не имеет права ожидать принятия запроса на W-канал, прежде чем утверждать WVALID. Нарушение этого требования может привести к взаимоблокировкам. Каждый бит в WSTRB соответствует одному 8-битному байту в WDATA. Если задан WSTRB, то записывается соответствующий байт в WDATA, в противном случае он игнорируется. WSTRB должен соответствовать AWSIZE и должен быть выровнен по AWADDR.

Таблица 5

Соответствие сигналов WSTRB и битов WDATA

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Биты в сигнале WSTRB | 3 | 2 | 1 | 0 |
| Соответствующие биты в WDATA | [31:24] | [23:16] | [15:8] | [7:0] |

Когда ведомое устройство готово, оно выдает сигнал AWREADY о том, что работа с данными в канале AW завершена . Ведомое устройство также выдает сигнал WREADY, сигнализирующий ведущему устройству о том, что запись данных принята . Затем шаги WVALID / WREADY повторяются для каждого такта в этой транзакции . При последнем такте мастер утверждает WLAST .

После последнего утверждения WVALID / WREADY с WLAST ведомый устанавливает сигналы на канале B, такие как: BID, BRESP, а затем устанавливает BVALID, чтобы сигнализировать ведущему, что ответ на запись выполнен . Мастер, когда все будет готово, выставляет сигнал BREADY, сигнализируя о завершении транзакции.

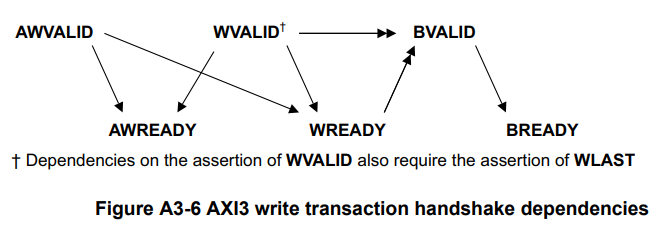


Рис.13 Последовательность состояний при операциях записи на шине AXI.

Одна из распространенных ошибок заключается в предположении, что ведущему устройству разрешено дождаться принятия AW, прежде чем вводить данные канала W . Это предположение неверно и может привести к взаимоблокировке. Для оптимальной производительности рекомендуется как можно раньше помещать данные канала AW и данные канала W.

После того, как по каналам AW и W принят последний такт, ведомому устройству разрешается подтвердить ответ сигнализации BVALID по каналу B. Канал B содержит сигнал BID, который должен соответствовать тому же значению, что и в запросе канала AW . BRESP соответствует его кодировке с помощью RRESP.

Ведущий должен размещать данные канала W в том же порядке, что и данные канала AW . Смешивание запросов между несколькими транзакциями по каналам AW и W запрещено.

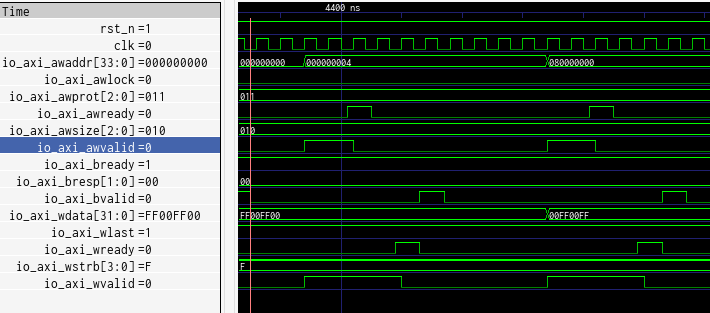


Рис. 14 Однотактная операция записи (INCR-wirte).



Рис. 15 Однотактная операция записи (INCR).

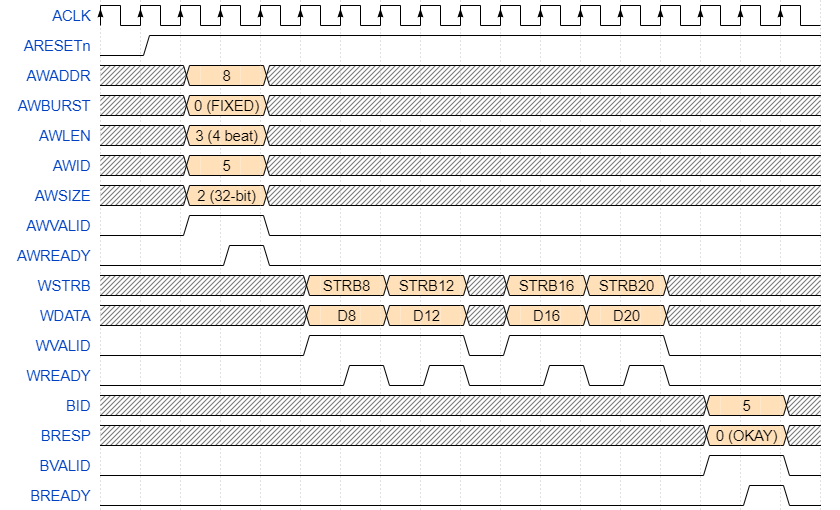


Рис.16 Четырехтактная операция записи (INCR).

Все современные системы-на-кристалле и все DRAM реализуют пакеты переноса для кэширования и конвейерного доступа к памяти, потому что делать это любым другим способом просто неэффективно. AXI4 требует определенных значений для сигнала AxLEN в пакетных запросах переноса, поскольку память SDRAM использует определенный фиксированный набор значений для длины пакета.

Для сравнения, фиксированные пакеты позволяют DMA и процессорам, использующим фиксированные пакеты, считывать/записывать несколько выборок данных в одно и то же местоположение. Это полезно, например, если у вас есть аудиосэмплы, которые считываются из памяти и передаются на аудиоконтроллер, который принимает данные по одному адресу.

Таблица 6

**Соответствие значений сигналов AxBURST и типов передачи**

|  |  |
| --- | --- |
| AxBURST[1:0] | Burst type |
| 0b00 | FIXED |
| 0b01 | INCR |
| 0b10 | WRAP |

**Идентификаторы транзакций и их использование**

Каждая транзакция на шине AXI4 может иметь идентификатор. Идентификатор передается ведущим устройством по каналу AW/AR и возвращается по каналам R и B. Обычно процессоры с внеочередным исполнением команд используют идентификатор для своих собственных нужд, например, для того, чтобы определить, какие данные регистра будут записаны обратно, или для сигнализации о различных потоках в одном ядре. Это позволяет процессору выполнять несколько запросов на чтение или запись, в то время как данные или ответ не поступают. Другой вариант использования - с помощью ригелей. Ригели увеличивают идентификационные биты на $c log2(количество мастеров). Это позволяет менеджеру узнать, на какое ведущее устройство перенаправлять ответ, затем биты идентификатора, которые использовались для этой цели, удаляются, и запрашивающему мастеру возвращается новый идентификатор, который соответствует идентификатору исходного запроса.

**AXI4-Lite**

AXI4-Lite был разработан для облегчения проектирования подчиненных устройств, которые не поддерживают пакетный доступ и идентификатор транзакции. Этот выбор позволяет разработчикам сократить площадь, используемую как ведомым, так и ведущим устройствами, и реализовать простые ведомые устройства, которые используют упрощенное соединение и их производительность не имеет значения, сохраняя при этом степень совместимости и обеспечивая простоту взаимодействия между AXI4 и AXI4-Lite. AXI4-Lite сокращает количество необходимых для взаимодействия сигналов – рис. 13.

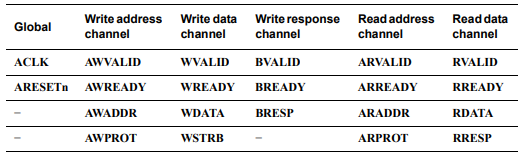


Рис. 17 Сигналы шины AXI4-Lite.

AXI4-Lite slave не может быть подключен к AXI4 master, который предоставляет пакетный доступ. Для этой задачи требуется специальный преобразователь. Однако, AXI4-Lite master можно подключить к AXI4 slave, подключив несуществующие сигналы к их значениям по умолчанию. Очевидно, что AXI4-Lite master может быть подключен к AXI4-Lite slave. Опять же, очевидно, что AXI4 master может подключаться к AXI4 slave.

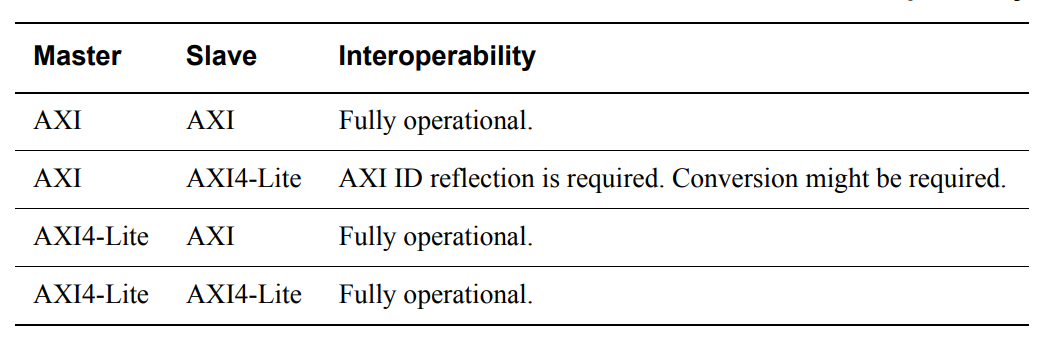


Рис. 18 Таблица совместимости AXI4 и AXI4-Lite.

**Протокол шины AMBA [AHB bus protocol]**

Спецификация шины AMBA AHB является частью спецификации шины, спецификация шины AMBA - это спецификация шины, предложенная компанией ARM, используемая в большинстве проектов С-н-К, которая обеспечивает AHB (Advanced High-performance Bus), ASB (Advanced System Bus), APB (Advanced Peripheral Bus).

AHB используется для подключения высокопроизводительной системной конфигурации с высокой тактовой частотой, для типичных приложений, таких как ядерная и высокоскоростная системная ОЗУ ARM, флэш-память NAND, DMA, мосты.

APB для подключения внешних устройств, в соответствии с требованиями к производительности, учитывает низкое энергопотребление.

ASB - это альтернатива AHB.

Преимущество шины AHB заключается в том, что она может быть подключена к микроконтроллеру (CPU), чипу оперативной памяти с высокой пропускной способностью, интерфейсу внешней памяти с высокой пропускной способностью, шине DMA master, имеет различные контроллеры интерфейса AHB и т.д. Вместе они образуют отдельную шинную систему, не только это, но и могут быть подключены с помощью системная шина APB AHB-мост APB. AHB может стать полноценным каркасом С-н-К.

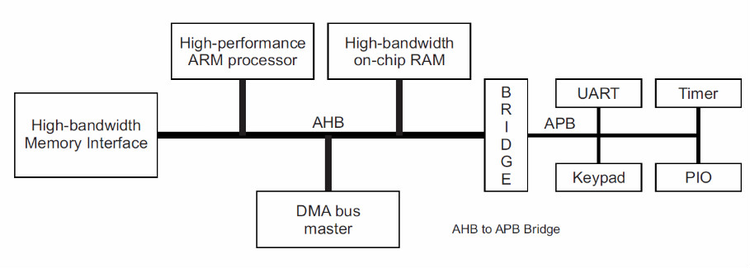


Рис.19 Типичная структура системы с шиной AHB.

AMBA AHB - это высокопроизводительный шинный интерфейс.   AMBA AHB реализует функции, необходимые для высокопроизводительных систем с высокой тактовой частотой, в том числе:

- пакетная передача данных;

- разделенная транзакции;

- однотактная передачи с передачей обслуживания;

- операции с одним тактовым фронтом;

- отсутствие Z-состояния на шине;

- поддержка «широких» шин данных (64/128 бит);

- конвейеризация обмена данными;

- устройство управления шиной может поддерживать несколько (до 16).

Наиболее распространенными подчиненными устройствами AHB являются устройства внутренней памяти, интерфейсы внешней памяти и периферийные устройства с высокой пропускной способностью.

Хотя периферийные устройства с низкой пропускной способностью могут выступать в качестве подчиненных устройств AHB, они обычно размещаются на расширенной периферийной шине AMBA (APB) из соображений производительности системы. Соединение между высокопроизводительными AHB и APB осуществляется с помощью AHB slave, называемого APB bridging.

Шина AHB содержит ведущие устройства (master), ведомые устройства (slave) и инфраструктурные устройства. К инфраструктурным устройствам относятся – арбитр, мультиплексор данных, блоки управления, мультиплексор адреса.

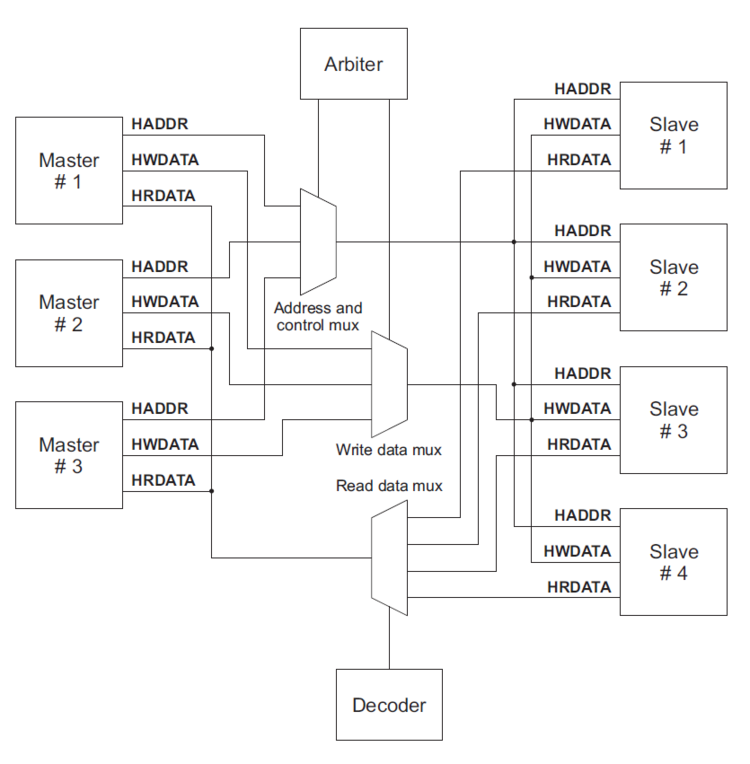


Рис. 20 Пример AHB шинной структуры

Ведущий( Master):

- инициирует операцию на шине (чтение/запись);

- допускается наличие только одного мастера на шине в текущий момент времени.

Ведомый( Slave):

- отвечает на запросы операций;

- выбирается по адресу, назначенному устройству.

Арбитр ( arbiter):

- гарантирует наличие одного ведущего на шине.

Дешифратор (decoder):

- определяет активные устройства в зависимости от их адресов.

Шину условно можно разделить на три составляющих:

- шина записи данных (HWDATA);

- шина чтения данных (HRDATA);

- шина контроля адреса (HADDR).

Таблица 7

Сигналы шины

|  |  |  |  |
| --- | --- | --- | --- |
| **Сигнал** | **Источник** | **Приемник** | **Описание** |
| HCLK | clock source | Все модули | Тактовый сигнал шины, тактирование по фронту |
| HRESETn | reset controller | Все модули | Сигнал сброса. Активный уровень – низкий |
| HADDR[31:0] | Master | decoder mux to slave arbiter | 32-битная шина адреса |
| HTRANS[1:0] | Master | mux to slave | Тип текущей транзакции NONSEQ, SEQ, IDLE, BUSY |
| HWRITE | Master | mux to slave | 1 - запись, 0 – чтение |
| HSIZE[2:0] | Master | mux to slave | Размер транзакции в байтах, до 1024 байт |
| HBURST[2:0] | Master | mux to slave | Тип пакета 4,8,16, с инкрементом / без |
| HPROT[3:0] | Master | mux to slave | Сигналы управления защитой, требуемые ведомым устройством с функцией защиты |
| HWDATA[31:0] | Master | mux to slave | Шина записи данных. Ведущий>ведомый |
| HRDATA[31:0] | Slave | mux to master | Шина чтения данных. Ведомый> Ведущий |
| HREADY | Slave | mux to master arbiter | Высокий уровень: ведомый показывает конец передачи  низкий: ведомому необходимо продлить цикл передачи |
| HRESP[1:0] | Slave | mux to master arbiter | Состояние передачи ведомого (OKAY, ERROR, RETRY, SPLIT) |
| HSELx | Decoder | slave | Сигнал выбора ведомого |

Таблица 8

Сигналы арбитража шины AHB

|  |  |  |  |
| --- | --- | --- | --- |
| Сигнал | Источник | Направление | Описание |
| HBUSREQx | Master | arbiter | запрос главного арбитра для получения сигнала запроса правильного использования шины, до 16 ведущих |
| HLOCKx | Master | arbiter | Если вы хотите быть ведущим в шине и во время передачи не хотите проиграть, вам нужно послать сигнал арбитру блокировки |
| HGRANTx | arbiter | master | Сигнал авторизации, текущий наивысший приоритет шины x. Когда HGRANTx и READY одновременно высокие, ведущий в праве опрашивать системную шину |
| HMASTER [3:0] | arbiter | slave has a split function | Arbiter for the ID assigned each a master, which pointed out that the main ongoing transmission equipment, split the information provided |
| HMASTLOCK | arbiter | slave has a split function | о указывает на то, что текущий мастер выполняет заблокированную операцию |
| HSPLITx[15:0] | slave | arbiter | От устройства, чтобы сообщить арбитру об этом сигнале, с помощью которого ведущее устройство, запущенное в очередной раз, попытается разделить передачи, каждая из которых соответствует ведущему устройству |

**Операции шины**

При возникновении необходимости воспользоваться шиной ведущий отправляет запрос арбитру, чтобы арбитр авторизовал указанный ведущий. Любой один период времени может быть в режиме master для доступа к шине, операции чтения и записи должны быть указаны ведомым. Начинается авторизованная передача по шине AHB, которая сначала отправляет адресные и управляющие сигналы, определяет направление передачи адресной информации, полосу пропускания и тип пакета. Какой из ведущих и ведомых декодеров адреса унифицированного планирования определяет адрес в сигналах передачи данных и управления. Передача данных завершается по шине данных. Чтобы избежать трехфазности шины, шина AHB разделяет чтение и запись, шину данных записи для передачи данных от ведущего устройства к ведомому устройству, шину данных чтения, используемую для передачи данных от ведомого устройства к ведущему устройству. И передача, включающая адрес каждого контрольного периода, один или более циклов передачи данных. Адрес и период управления не могут быть увеличены, и поэтому сигнал адреса подчиненного устройства должен выбираться в цикле.

Данные HREADY циклически передаются путем распространения сигнала, но к состоянию ожидания передачи добавляется значение HREADY, которое является низким, так что требуется дополнительное ведомое устройство или время для предоставления выборочных данных, отражающих дополнительное состояние передачи с помощью ответного сигнала HRESP ведомого устройства. при обычных обстоятельствах для завершения полной пакетной передачи данных у arbiter будет лицензия на доступ к другой основной шине, однако, чтобы избежать чрезмерной задержки, arbiter может прервать пакетную передачу. В этом случае необходимо снова подключить ведущую шину к оставшейся части пакетной передачи.

**Базовая передача**

Передается суммой следующих двух компонентов:

Фаза адреса: период

Фаза передачи данных: один или более циклов, для действительного периода требуется несколько пакетов сигналов, период может быть продлен запросом, который уже готов.

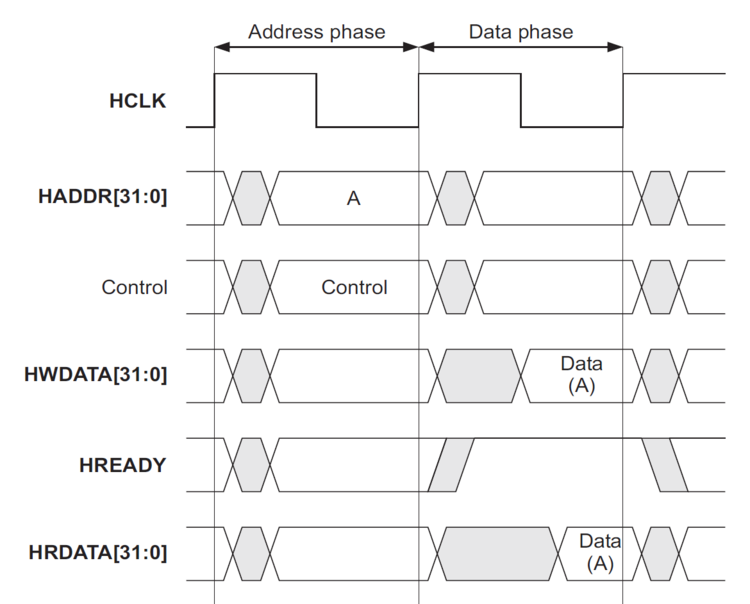


Рис. 21 Передача без циклов ожидания.

По фронту первого такта ведущий передает адресные и управляющие сигналы; По фронту второго такта, адресные и управляющие сигналы ведомого устройства отбираются и выводятся в готовом виде.

Если выполняется операция записи данных, ведущий передаст данные для записи по фронту второго такта. Если это операция чтения, данные ведомого устройства считываются после того, как шина записи выставит сигнал HREADY.

Третий такт, по фронту:

– если это операция записи, ведущий выставляет сигнал HREADY, что данные ведомого устройства были успешно получены, что свидетельствует об успешном завершении операции;

- если операция представляет собой считывание, то главный сигнал приема HREADY, указывающий на то, что считанные данные действительны и получены и означает, что операция выполнена успешно.

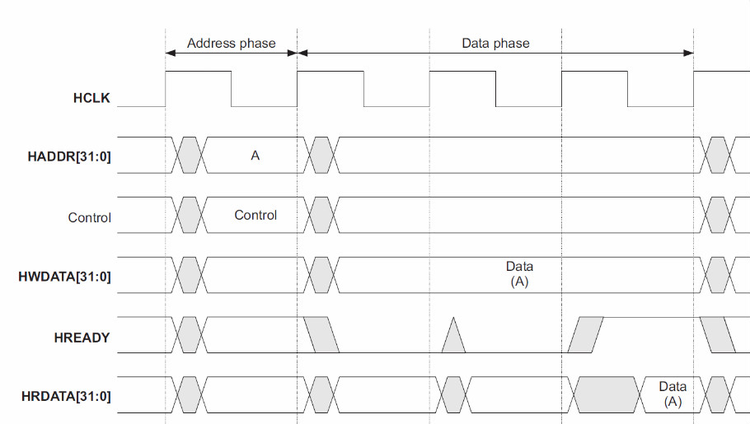


Рис. 22 Одиночная передача с тактом ожидания, внесенного ведомым.

Передача со множеством конвейеризированных операций

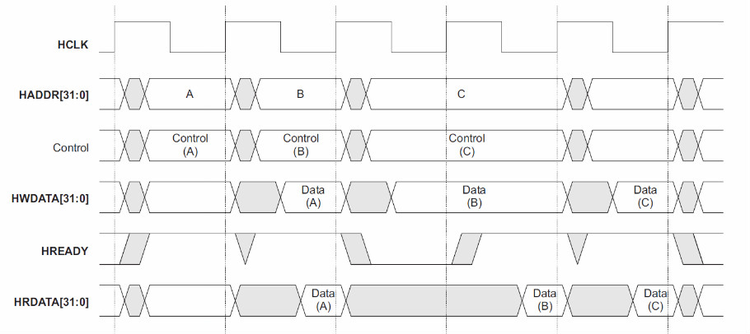


Рис. 23 Передача со множеством конвейеризированных операций.

В первом цикле ведущий инициирует процедуру A и подает сигналы адреса и управления. Во втором цикле ведомый получает запрос от шины (сигнал HREADY высокий). После начала второго цикла мастер обнаруживает необходимость выполнения операции B и проверяет готовность предыдущего цикла, инициируя вторую операцию B.

Третий цикл, при получении сигнала готовности мастера, означает, что операция A завершена. После начала третьего цикла ведущий обнаружил, что C нуждается в выполнении операций, и проверяет, что время выполнения предыдущего цикла велико.

После окончания третьего цикла ведомый занят из-за состояния ожидания. Четвертый цикл - получение низкого сигнала HREADY ведущим, ведомое устройство знает, что он хочет подождать, а затем удерживает ведущее устройство.

Четвертый цикл, ведомое устройство завершило обработку транзакции, сигнал HREADY высокий, и можно продолжать обработку. На пятом цикле, когда сигнал ведущего HREADY является высоким, процесс B, возможно, уже знает о подчиненной операции. После начала пятого цикла операция B завершена. После окончания шестого цикла операция C завершена.

**Инкрементная пакетная передача**

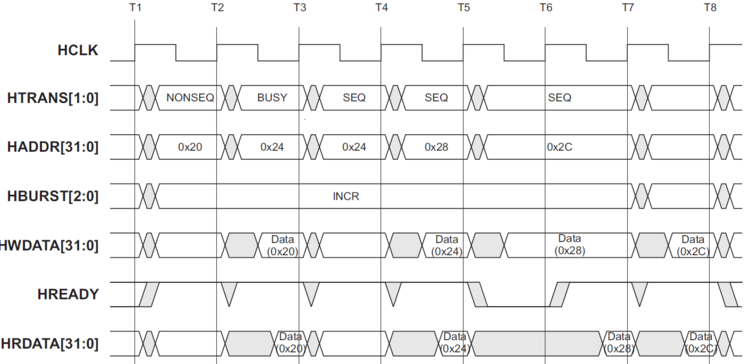


Рис. 24 Инкрементная пакетная передача.

T1, мастер принимает входящие адресные и управляющие сигналы, поскольку запускается новая серия, передача типа NONSEQ.

T2, поскольку мастер не может обработать вторую серию во втором цикле, мастер использует передачу BUSY, чтобы продлить период времени для себя. Обратите внимание, что, хотя это и длительный период, ведущему устройству необходимо предоставить второй адрес передачи и управляющие сигналы.

T3, ведомое устройство для основной коллекции отправлено BUSY, знайте, что нужно дождаться сигнала от ведущего устройства, чтобы ведомое устройство проигнорировало эту передачу BUSY.

T4, ведущее устройство инициирует вторую передачу, как и вторую передачу одиночного пакета, поэтому тип последующей передачи будет изменен.

T5, ведомое устройство уже подаст сигнал "низкий", сообщая ведущему устройству о необходимости подождать некоторое время.

T8 - время завершения последней передачи.

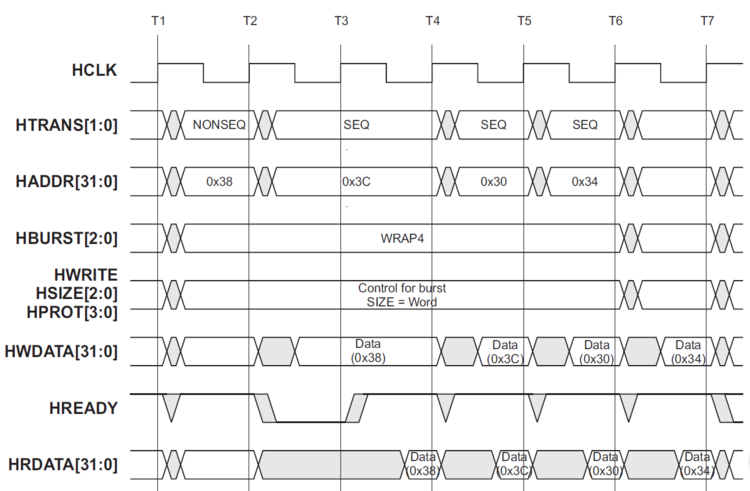


Рис. 25 4-тактная передача пакета.

Единственное отличие состоит в том, что раньше при использовании разных адресов, после адреса 0x3C, граничного цикла адреса, третий такт адреса становится равным 0x30. Всего четыре такта, четыре адреса, каждый из которых отличается, что составляет диапазон адресов с четырьмя циклами. Он определяет начальный адрес рабочего диапазона обратной связи в цикле.

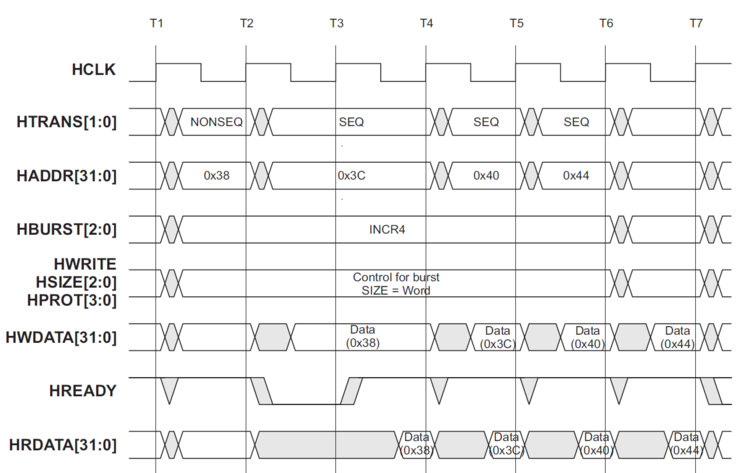


Рис. 26 Инкрементная передача

И в основном то же самое, что описано выше, но не адрес не зацикливается на 0x30, а увеличивается с шагом.

**Управляющие сигналы**

|  |  |  |
| --- | --- | --- |
| **HTRANS[1:0]** | **Тип** | **Описание** |
| 00 | IDLE | Занимает ведущее устройство шины, но не используется для передачи данных. В это время, даже если ведомое устройство включено, оно не получает никакого сигнала от шины данных. В это время, если выбрано значение salve, при каждом простое прохождении цикла slave HRESP [1: 0] возвращает ответ OKAY |
| 01 | BUSY | Занято ведущее устройство шины, но процесс пакетной передачи еще не готов к следующей передаче Промежуточное хост-устройство пакетной передачи может отправить сообщение BUSY. На этом ведомом устройстве не принимаются данные с шины, а выполняется ожидание, и по сигналу [1: 0] возвращается ответ "OKAY". Следует отметить, что эта передача нуждается в адресации и подаче управляющих сигналов для следующего кадра, хотя ведомое устройство не будет выполнять выборку. |
| 10 | NONSEQ | Для указания времени первой передачи отдельных данных или пакета данных Время первой передачи независимо от адреса и управляющих сигналов. |
| 11 | SEQ | При передаче следующего пакета данных адрес и адрес последней передачи являются релевантными, за определенное время до подачи управляющего сигнала на шину адрес должен быть увеличен или, при необходимости, зациклен. |

|  |  |  |
| --- | --- | --- |
| **HSIZE[2:0]** | **Size** | **Description** |
| 000 | 8 bits | Byte |
| 001 | 16 bits | Halfword |
| 010 | 32 bits | Word |
| 011 | 64 bits | - |
| 100 | 128 bits | 4-word line |
| 101 | 256 bits | 8-word line |
| 110 | 512 bits | - |
| 111 | 1024 bits | - |

Пакетная передача – пакет не должен пересекать границу 1К

|  |  |  |
| --- | --- | --- |
| **HBURST[2:0]** | **Types of** | **Description** |
| 000 | SINGLE | Single transfer |
| 001 | INCR | Incrementing burst of unspecified length |
| 010 | WRAP4 | 4-beat wrapping burst |
| 011 | INCR4 | 4-beat increment burst |
| 100 | WRAP8 | 8-beat wrapping burst |
| 101 | INCR8 | 8-beat increment burst |
| 110 | WRAP16 | 16-beat wrapping burst |
| 111 | INCR16 | 16-beat increment burst |

**Ответный сигнал**

После того, как ведущее устройство инициирует суммарную передачу, ведомое устройство может определить ход этой передачи, и ведущее устройство передачи не может отменить уже выполненную передачу. Сигнал завершения готовности ведомого устройства, передаваемый через сигналы HRESP [1: 0], отражает состояние передачи. ведомое устройство может завершить передачу следующим образом:

- немедленно завершить суммарную передачу;

- один или несколько циклов задержки для завершения передачи;

- ошибка возврата при передаче не удалось;

- задержка передачи приводит к отключению шины.

Передача завершается сигналов HREADY.

Когда он высокий передача завершена, когда низкий дополнительная задержка.

Коды HRESP [1: 0]:

00: OKAY

01: ERROR

10: RETRY (предача не завершена, запрос на передачу возобновлен главным устройством, арбитр продолжит использовать обычный приоритет)

11: SPLIT

**Декодирование адреса**

Декодер адреса для обеспечения сигнала выбора HSELx для каждой ведомой шины, сигнал выбора генерируется путем комбинирования логического адреса с кодом декодирования. Только после того, как текущая передача данных будет завершена (значение HREADY будет высоким), будут выбраны адрес ведомого устройства и управляющие сигналы, а также значение HSELx. В этом случае это может произойти при определенных условиях: при генерации сигнала HSELx значение HREADY будет низким, значение slave изменится после текущей передачи. У каждого ведомого устройства наименьшее адресное пространство - 1 Кбайт, ограничение на пакетную передачу у ведущего устройства составляет 1 Кбайт, что позволяет избежать трансграничных проблем. Когда проект не будет использовать все адресное пространство, может появиться доступ для устранения несуществующей ситуации, что требует добавления дополнительного подчиненного устройства по умолчанию для обеспечения ответа на вышеуказанную ситуацию. Когда адрес доступа к передаче, не соответствующий стандарту или SEQ, не существует, должен быть выдан ответ об ошибке ведомого устройства по умолчанию; когда доступ к адресу, указанному в режиме BUSY или ожидания, не существует, ведомое устройство по умолчанию отвечает нормально. Декодер адресов может работать с подчиненной функцией по умолчанию.

**Арбитраж**

Механизм арбитража гарантирует, что в любой момент времени доступ к шине может получить только ведущий сервер. арбитр определяет, какой запрос на доступ к шине может быть выдан для доступа к шине, это достигается с помощью алгоритма приоритета. Спецификация AHB не содержит алгоритма приоритета, разработчикам необходимо определить конкретные системные требования. Как правило, арбитр, не прерывая пакетную передачу, передает право доступа к шине другому ведущему. Конечно, пакетная передача неопределенной длины прерывается, это зависит от того, как задан алгоритм приоритета. 4. Если пакет данных прерывается, когда мастер снова получает доступ, оставшаяся часть передается. Сумма длительности передачи в передаче INCR8 прерывается через 3 такта, мастер снова получает авторизацию доступа, мы продолжаем передачу в течение оставшихся 5 тактов, оставшаяся часть может состоять из ОДНОГО INCR4 и композиции, или из INCR. HBUSREQx: ведущий запрашивает доступ к сигналам арбитра. HLOCKx: Указывает, не может ли быть выполнено прерывание передачи, в то время как сигнал отправляется от ведущего к арбитру и HBUSREQx. HGRANTx: генерируется сигнал главного арбитра, указывающий на авторизацию, когда сигнал высокий, а HGRANTx HREADY высокий, ведущий сервер шины может передавать сигнал по указанному адресу. HMASTER [3: 0]: генерируется главный арбитр, который указывает на авторизацию, сигнал управления адресом для мультиплексора, позволяющий выбрать доступ к главному шине. HMASTERLOCK: генерируется арбитр, указывающий, является ли текущая передача последовательностью блокировки передачи. HSPLIT: поддержка использования разделения передачи

**Шина Wishbone**

Шина обычно используется для подключения центрального процессора и периферийных устройств.

Простейшая версия Wishbone - Wishbone Classic Standard.

Сигналы шины:

1. CLK\_I: тактовый вход;
2. STB\_O: высокий уровень означает, что мастер хочет отправить запрос;
3. ACK\_I: Высокий уровень означает, что подчиненное устройство обрабатывает запрос;
4. ADR\_O: адрес, который мастер хочет прочитать или записать;
5. WE\_O: независимо от того, хочет ли мастер читать или записывать;
6. DAT\_O: данные, которые мастер хочет записать;
7. SEL\_O: разрешение чтения / записи байтов мастера;
8. DAT\_I: данные, считываемые ведущим устройством из ведомого устройства;
9. CYC\_O: включить сигнал шины.

Пример временных диаграмм работы Wishbone Classic Standard

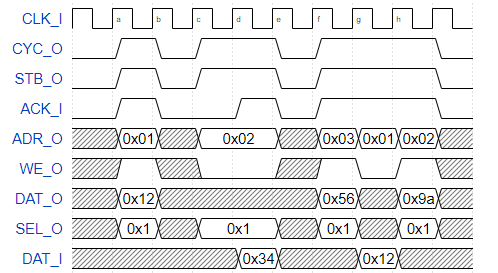


Рис. 27 Пример временных диаграмм работы Wishbone Classic.

**Классический конвейерный протокол Wishbone**

Приведенный выше стандартный протокол Wishbone Classic очень прост, но он сталкивается с проблемой: предположим, что реализацией является контроллер SRAM, который имеет задержку чтения на один цикл, т. е. адрес задается за один цикл, а результат недоступен до следующего цикла. В стандарте Wishbone Classic будет получаться следующее – рис.:

- а) ведущий выдает адрес чтения 0x01, затем контроллер SRAM начинает считывание, но данные еще не считаны обратно, поэтому ACK\_I=0.

- б) SRAM завершает чтение, вводит прочитанные данные 0x12 в DAT\_I и устанавливает ACK\_I=1.

- в) ведущий выдает следующий адрес чтения 0x02, SRAM должен начать чтение снова;

- г) SRAM завершает второе чтение, помещает прочитанные данные 0x34 в DAT\_I и устанавливает ACK\_I=1.

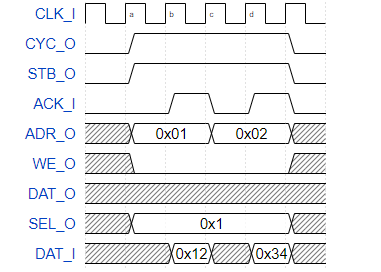


Рис. 28 Временные диаграммы Wishbone Classic при работе с контроллером памяти.

Особых с функционированием нет, но каждые два цикла может выполняться только одна операция чтения, что не лучшим образом сказывается на производительности.

Как решить эту проблему? Мы указываем первый адрес на этапе а) и получаем первые данные на этапе б), затем, если мы можем указать второй адрес нга этапе б), мы можем получить вторые данные на этапе в). Таким образом, конвейерная операция чтения может выполняться один раз за такт. Для корректной работы такого конвейерного запроса протокол необходимо немного изменить.

Идея проста: поскольку стандарт Wishbone Classic считает, что первый запрос находится на рассмотрении в цикле б), пусть первый запрос будет выполнен ранее в цикле а), за исключением того, что его данные будут недоступны до цикла б). Фактически, операцию чтения в это время можно рассматривать как разделенную на две части: сначала ведущее устройство отправляет запрос на чтение подчиненному устройству, который выполняется в цикле а); затем ведомое устройство отправляет результат чтения ведущему устройству, который выполняется в цикле б). Изменения следующие:

* добавляется сигнал STALL\_I : CYC\_O=1 && STB\_O=1 && STALL\_I=0 – показывает запрос на чтение;
* изменяется семантика сигнала ACK\_I : CYC\_O=1 && STB\_O=1 && ACK\_I=1 ответ на запрос чтения.

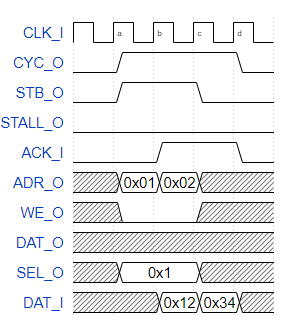


Рис. 29 Временные диаграммы двух последовательных циклов чтения шины Wishbone Classic Pipelined bus.

В итоге протокол работы Wishbone Classic Pipelined следующий:

- а) ведущий запрашивает чтение адреса 0x01, ведомый получает запрос на чтение (STALL\_O=0);

- б) ведомый возвращает результат запроса на чтение 0x12 и устанавливает ACK\_I=1; в то же время, когда ведущий запрашивает адрес чтения 0x02, подчиненное устройство получает запрос на чтение (STALL\_O=0)

- в) ведомое устройство возвращает результат запроса на чтение 0x34 и устанавливает ACK\_I=1; ведущее устройство больше не инициирует запрос и устанавливает STB\_O=0

- г) все запросы завершены, выставляем CYC\_O=0

Таким образом, мы ведомое устройство, которое выполняет одну операцию чтения за цикл, пусть и со сдвигом по времени на такт.

**Заключение**

Как видно, из рассмотренных шин самой простой является Wishbon, и именно поэтому она столь популярная в проектах с открытым кодом.

Шины AXI, AHB безусловно являются наиболее продвинутыми и комплексными, обеспечивая и арбитраж, и различные типы передачи. Данные шины больше подходят для достаточно крупных проектов, ориентированных на интеграцию с аналогичными модулями, или в процессорную систему с аппаратными процессорами (как в сериях Zynq или старших сериях Cyclone V). Обычной практикой при работе с AXI/AHB является включение в проект IP-ядер с указанными интерфейсами в проекты – как правило, фирменные среды разработки для FPGA предоставляют достаточно широкий выбор, даже в свободном доступе без лицензирования.

**Литература**

1. <https://adaptivesupport.amd.com/s/article/1053914?language=en_US>
2. <https://support.xilinx.com/s/topic/0TO2E000000YNxCWAW/axi-basics-series?tabset-50c42=2>
3. <https://habr.com/ru/articles/572926/>
4. <https://www.codetd.com/en/article/15908184>
5. <https://www.sobyte.net/post/2022-06/wishbone/>
6. <https://programmersought.com/article/50562238743/>