**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**

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.

**Каналы**

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.

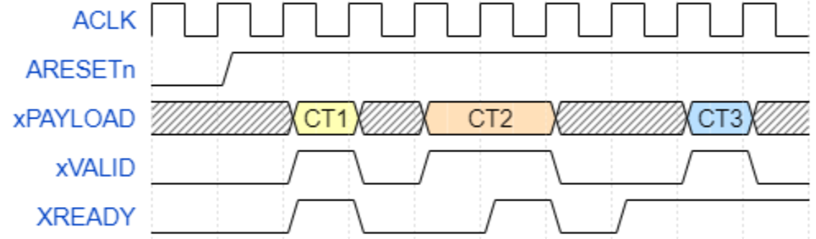


Рис.1 Пример трех циклов транзакций.

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).

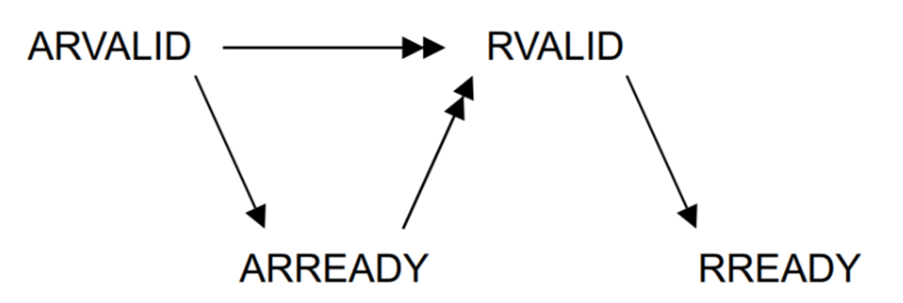


Рис. 2 Последовательность состояний при запросе на чтение.

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.

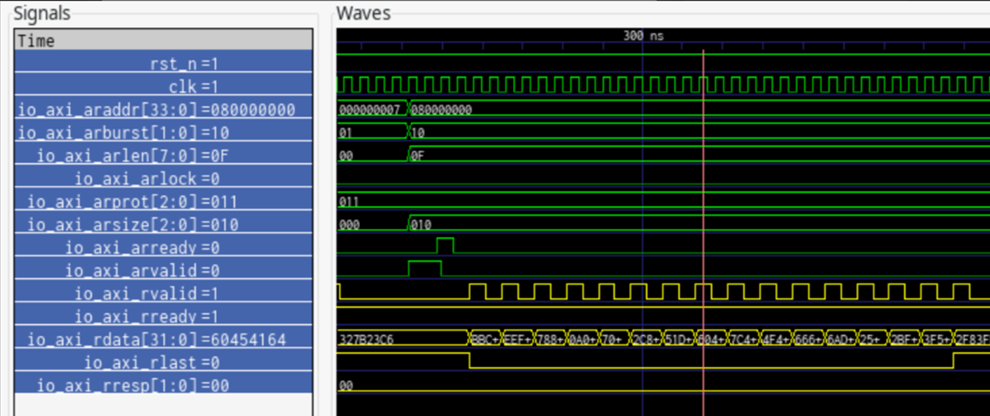


Рис. 3 Временные диаграммы при транзакции на AXI.

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.

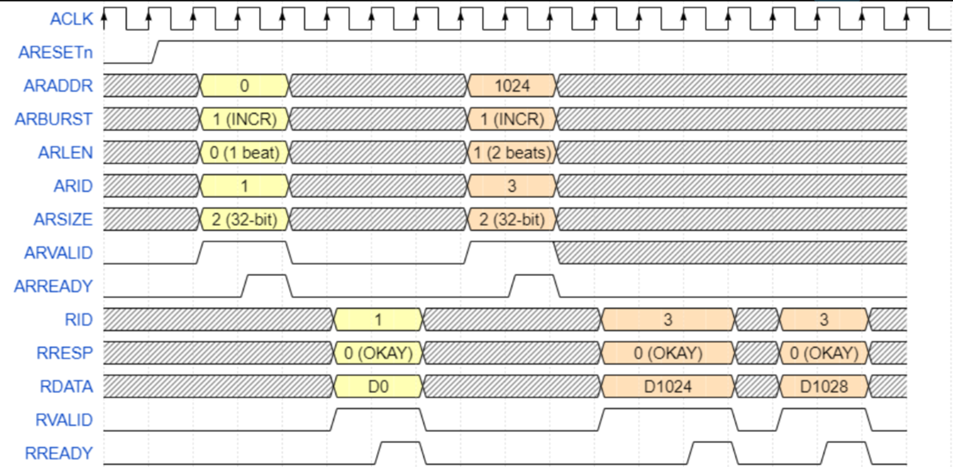


Рис. 4 Светло-желтым цветом – первая транзакция, представляет собой пакет длиной в один такт. Вторая - пакет длиной в два такта.

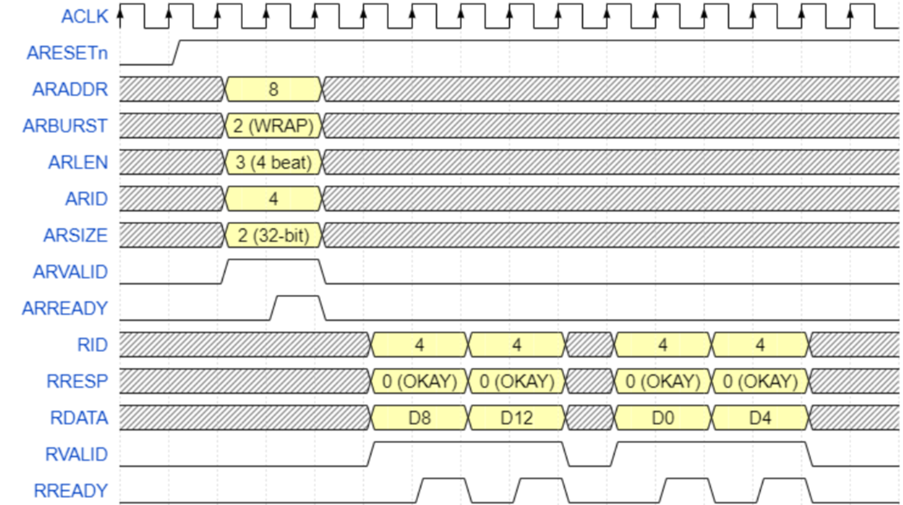


Рис. 5 Временная диаграмма сигналов в 4-тактной транзакции (тип beat).

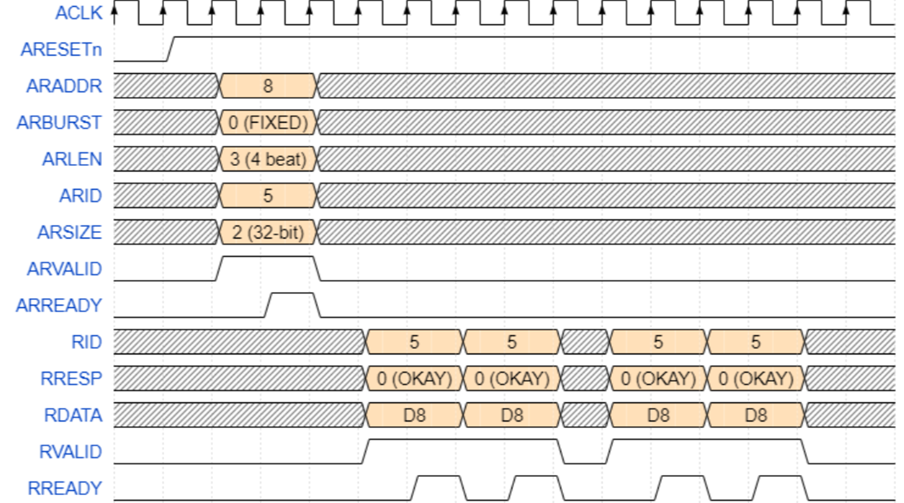


Рис. 6 Временные диаграммы 4-тактной транзакции (тип Fixed).

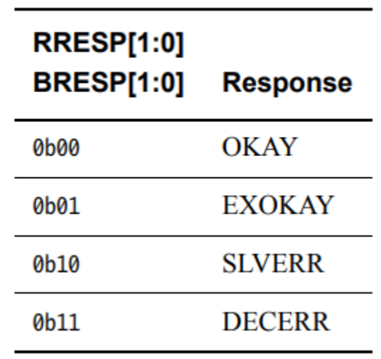


Рис.7 Значения сигналов RRESP/BRESP и их семантика.

Подчиненные устройства могут выбрать, принимать ли запись или чтение в первую очередь, или и то, и другое, однако после передачи ответа на запись в память шины B подчиненные устройства должны отвечать обновленным значением для каждого считывания. Interconnect и crossbar используют сигнал xLAST для различения нескольких сигналов, interconnect расширяет идентификационные биты, чтобы содержать информацию об отправителе запроса, и когда данные возвращаются, те же идентификационные биты используются для принятия решения о том, на какой адрес должен быть направлен основной ответ.

Одна из распространенных ошибок разработчиков заключается в предположении, что сигнал RRESP канала R остается неизменным между несколькими переходами сигнала beat в одном и том же пакете. Это неверно, каждый beat может содержать различный отклик. Вторая наиболее распространенная ошибка заключается в предположении, что для подтверждения RVALID не требуется подтверждать ARREADY. Это не так, как описано в разделе "Взаимосвязи между каналами" в спецификации AXI4. Третья наиболее распространенная ошибка заключается в предположении, что если запущено несколько AR-транзакций, они будут завершены в том же порядке, что и запрошенные, если у них разные идентификаторы. Это не так, как описано в разделе "Взаимосвязи между каналами". Подчиненные устройства должны сохранять несколько транзакций с одним и тем же идентификатором в том же запрошенном порядке, но это требование недействительно для транзакций с разными идентификаторами.

**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.

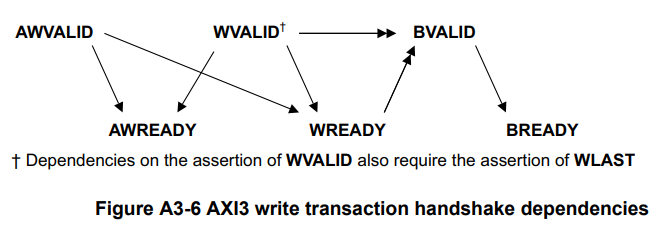


Рис.8 Последовательность состояний при операциях записи на шине AXI.

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.

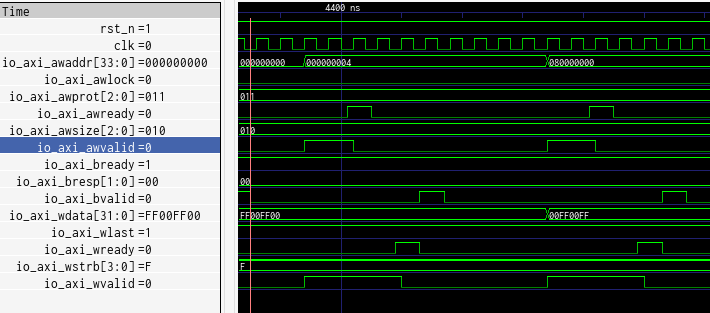


Рис. 9 Однотактная операция записи (INCR-wirte).



Рис. 10 Однотактная операция записи (INCR).

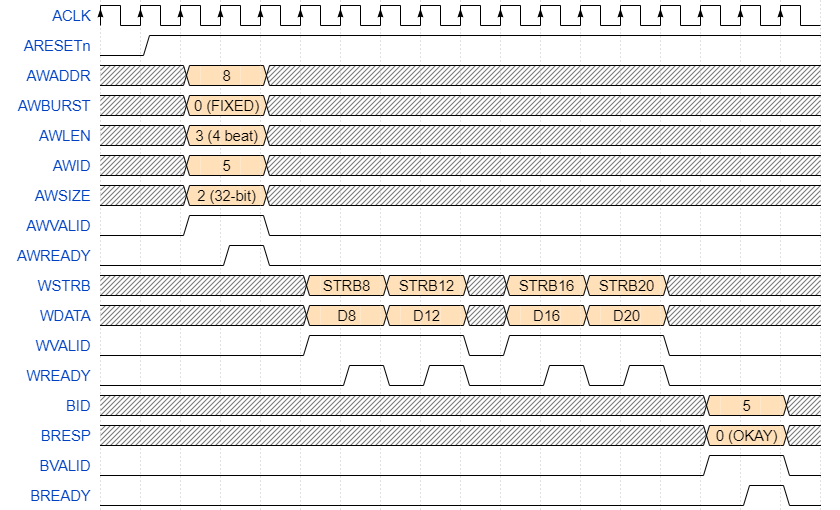


Рис.11 Четырехтактная операция записи (INCR).

Цс

**Ширина сигнала**

Существуют параметры, определяющие ширину некоторых сигналов: ширина данных, ширина адреса и ширина идентификатора.

Обычно используется фиксированную ширину данных и настраиваемую ширину адреса / идентификатора . Иногда требуется, чтобы биты адреса состояли из некоторого количества битов в подчиненной реализации .

Обмен данными по ширине осуществляется преобразователями ширины шины данных, а AXI4 реализует сигналы, которые используются для облегчения этой формы обмена данными . Одним из этих сигналов является AxSIZE, однако реализация должна соответствовать правилам для битов AxCACHE.

Большинство реализаций просто добавляют нули для расширения и отбрасывают ненужные биты для уменьшения ширины адреса. Выбрасывать биты адреса настоятельно не рекомендуется, поскольку это может привести к недоступности некоторых диапазонов адресов.

Диалог с уменьшением ширины идентификатора неразумен, но его легко реализовать, сопоставив несколько идентификаторов одному и тому же идентификатору и затем запомнив, в каком порядке был отправлен запрос на этот идентификатор . Когда приходит ответ, мы отменяем сопоставление узкого идентификатора с широким идентификатором, потому что мы знаем, что для того же идентификатора порядок транзакций сохраняется .

Запрос ID up выполняется просто путем добавления нулей слева. Обычно interconnect и crossbar увеличивают ширину идентификатора, чтобы включить информацию об источнике транзакции . Когда поступает ответ, этот идентификатор затем используется для перенаправления ответа ведущему устройству, инициировавшему транзакцию .

Сигналы ARCACHE/AWCACHE

Сигналы ARCACHE и AWCACHE используются, чтобы дать cache / crossbar и interconnect представление о требованиях к транзакции: распределении кэша, буферизации, политике модифицируемости .

Адресная и пакетная логика

AXI4 реализует пакетные транзакции для повышения производительности и реализует несколько активных транзакций для повышения производительности в системах с высокой задержкой . Транзакции чтения и записи инициируются соответственно по каналам AR / AW . Каналы AR и AW содержат четыре сигнала, используемых в генерации адресов и логике пакетной обработки: AxADDR, AxSIZE, AxLEN, AxBURST.

AxADDR содержит начальный адрес, однако для некоторых типов пакетов он может не соответствовать размеру шины. Все адреса представляют собой значения, выровненные по байтам .

AXI4 определяет следующее: одиночный такт - это данные на канале W или R-канале с момента утверждения VALID до момента утверждения READY. По шине W / R может быть передано несколько тактов, количество этих тактов определяется значением AxLEN плюс один. Последний такт всегда подтверждает сигнал xLAST, сигнализирующий соединению о том, что был выполнен одиночный пакет . Одиночный пакет определяется как все такты от первого до последнего с утвержденным сигналом xLAST. Одна транзакция содержит один бит адреса и AxLEN + 1 бит данных . Для операций записи сюда также входит один бит в канале B, который содержит сигналы RESP и ID.

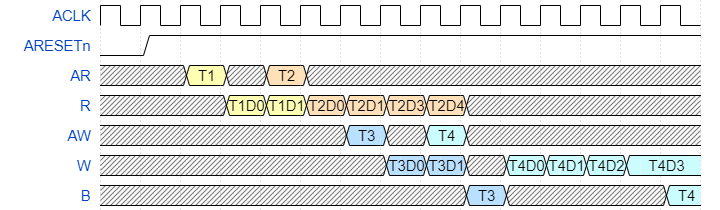


Рис. 12 Пример нескольких транзакций: 1 (arlen=1), T2 (arlen=3) транзакции чтения и T3 (awlen=1), T4 (awlen=3) транзакции записи.

Сигнал AxSIZE содержит значение log2 для количества байтов, которые будут переданы за один такт . AxSIZE может достигать максимального размера шины данных . AxSIZE не является обязательным для ведущего устройства, и если он не реализован, то используется значение, соответствующее ширине шины данных. Подчиненные устройства должны реализовать этот сигнал. AxSIZE не изменяет количество бит в одной транзакции, но вместо этого указывает ширину одного бита .

Maxlen определяет количество тактов в одной транзакции минус один. Например, значение, равное нулю, означает, что имеется один такт. Этот сигнал не является обязательным для ведущих устройств, и если ведущий не выполняет его, то на вход ведомого устройства присваивается значение 0 (одиночный такт). AxBURST управляет генерацией внутренних адресов и позволяет выбирать между FIXED - одинаковым адресом для всех тактов, INCR - адрес каждого такта увеличивается на 1 << AxSIZE и WRAP - адрес каждого такта увеличивается на 1 << AxSIZE, но адреса переносятся.

Другими словами, первый такт начинается с адреса, указанного AxADDR и увеличивается на 1 << AxSIZE, однако при переключении битов, соответствующих значению AxLEN, они приводят к тому, что те же биты следующих адресов начинаются с 0. Для пакетов переноса требуется, чтобы AxLEN имел любое значение: 1, 3, 7, 15. Для всех пакетов адреса должны быть выровнены в соответствии с размером AxSIZE.

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.

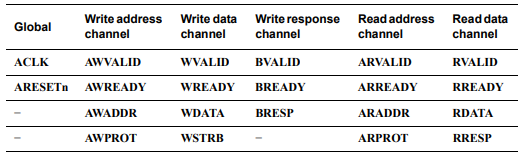


Рис. 13 Сигналы шины AXI4-Lite.

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.

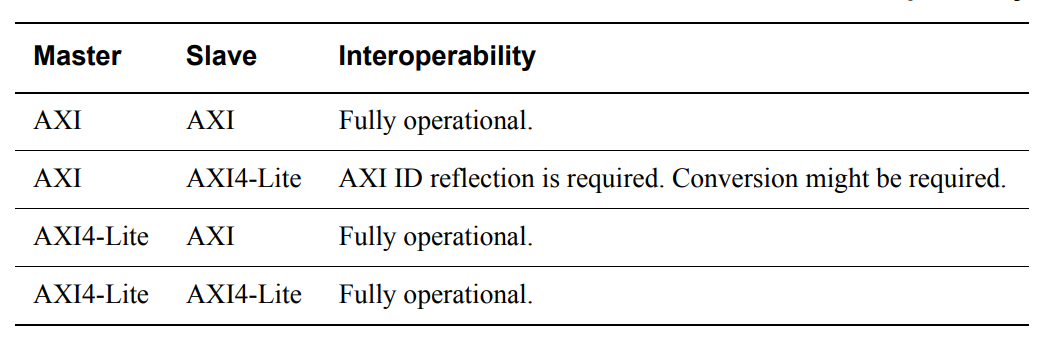


Рис. 14 Таблица совместимости AXI4 и AXI4-Lite.

Чтобы понять, почему в AXI4 используются каналы, нам нужно понять, что такое регистровые блоки, как разделение сигналов на каналы помогает при проектировании регистровых блоков и почему используется регистровый блок. Давайте разберемся, что ограничивает максимальную частоту схемы. Давайте взглянем на абстрактную структуру межрегистровой логики.

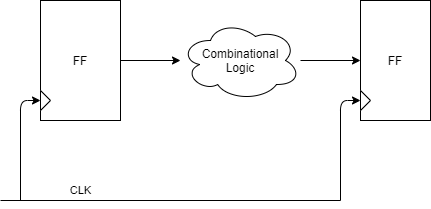


Рис.15 Блоки логики между регистрами.

Людям, имеющим опыт в разработке RTL, эта схема хорошо знакома. Первый триггер запускает выходные данные с положительным фронтом тактовой частоты clk. Данные проходят комбинационную логику, и через время задержки T стабилизируются на входах второго триггера. Если период тактового импульса меньше времени прохождения сигнала через комбинационную логику, то второй триггер будет фиксировать неверные данные.

Одним из параметров, которым можно управлять, является комбинационная логика. Уменьшая задержку в схеме комбинационной логики, увеличиваем максимальную частоту схемы. Например, разделяя комбинационную логику на более мелкие части, которые имеют меньшие задержки, можно увеличить максимальную частоту, но при этом увеличивается время до появления сигнала на выходе (т.н. латентность).

В AXI4 принято использовать регистровые блоки (register slices). Регистровые блоки содержат регистры-защелки, фиксирующие данные канала. Фиксируя выходные сигналы, мы гарантируем, что комбинационная задержка одной схемы не повлияет на комбинационную задержку другой схемы. Существует два типа регистровых блоков. Упрощенный с задержкой в 1 цикл и 50%-ным использованием канала и полностью регистровый, который имеет ту же задержку в 1 цикл, но не страдает от 50%-ного использования канала. Упрощенный фрагмент регистра обычно используется для каналов AW/AR/B.

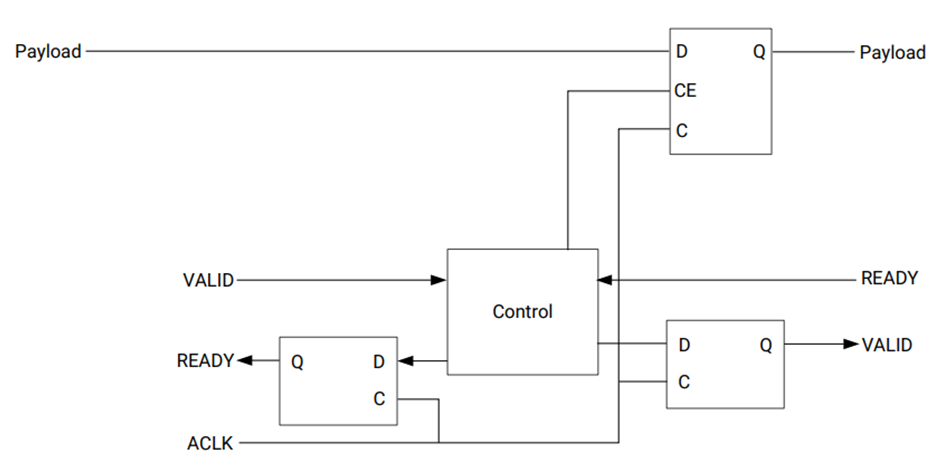


Рис. 16 Структура упрощенного регистрового блока.

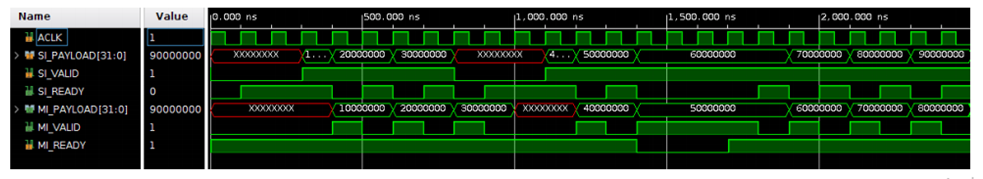


Рис. 17 Временные диаграммы работы регистрового блока.

Полностью регистровый блок реализует два узла для хранения данных. Это улучшает функциональность, но приводит к увеличению занимаемой площади, поскольку требует удвоенного количества триггеров. Если в промежуточном триггере не сохранено никаких данных, то данные пересылаются на вывод непосредственно с ввода. Если выходной сигнал в данный момент занят, то вход фиксируется в промежуточном триггере, а затем выходной сигнал, наконец, становится свободным, данные с промежуточного триггера передаются на выходной триггер. Таким образом, не требуется циклический переход и достигается 100%-ная загрузка канала. По этой причине он обычно используется для каналов чтения-записи (W/R-каналы).

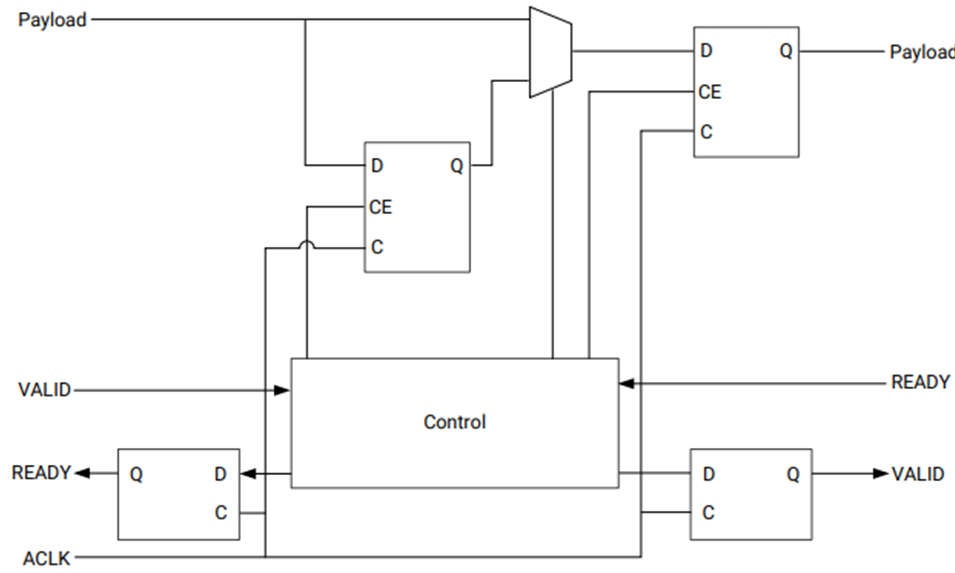


Рис. 18. Структура полного регистрового блока.

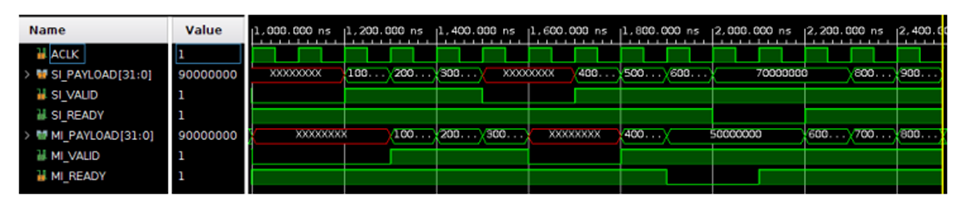


Рис. 19. Временные диаграммы работы регистрового блока.

**Синхронизация**

Одно из преимуществ использования AXI4 в качестве шины систем-на-кристалле заключается в том, что для синхронизации отдельных блоков можно использовать простые асинхронные FIFO. Для С-н-К характерно наличие нескольких доменов синхронизации.

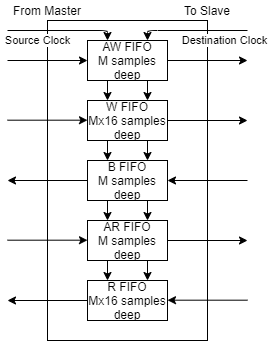


Рис. 20 Взаимодействие различных тактовых доменов на основе FIFO.

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

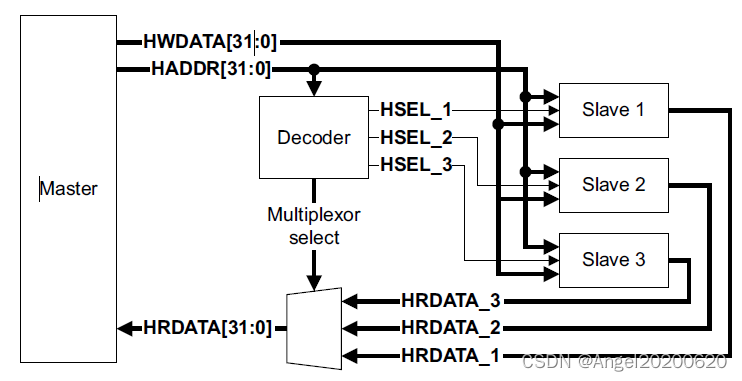
AMBA AHB - это высокопроизводительный шинный интерфейс.   AMBA AHB реализует функции, необходимые для высокопроизводительных систем с высокой тактовой частотой, в том числе:

* Пакетная передача;
* Работа с одним тактовым фронтом;
* Ширина данных в битах поддерживает 64, 128, 256, 512 и 1024.

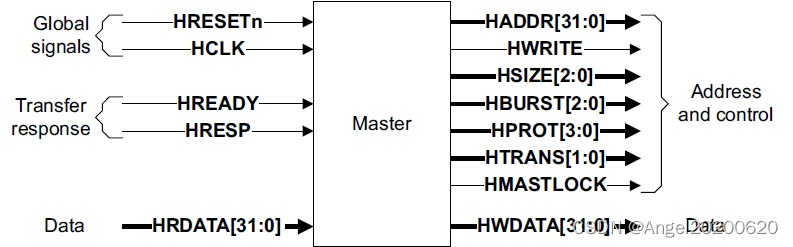
Наиболее распространенными подчиненными устройствами AHB являются устройства внутренней памяти, интерфейсы внешней памяти и периферийные устройства с высокой пропускной способностью.

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.

На рисунке 1 показан ведущий модуль AHB, подключенный к 3 ведомым устройствам AHB. Логика соединения шин включает модуль декодирования адреса и мультиплексор "подчиненный -ведущему". Модуль декодирования адреса преобразует адрес, отправленный ведущим устройством, в подходящий сигнал SEL При выборе ведомого устройства мультиплексор разумно выбирает выходные данные каждого ведомого устройства для ведущего устройства. Обратите внимание, что шина AHB может иметь несколько ведущих устройств.



**Ведущее устройство (master)**

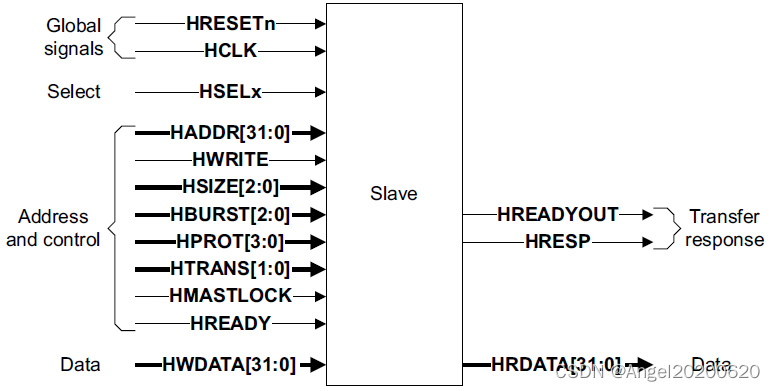
Возьмем типичный мастер в качестве примера для представления интерфейсных сигналов.  
  
INPUT信号：

* **HREADY** : It is given by Slave, indicating that the transmission is complete. The next transfer is ready.
* **HRESP** : Some additional status information given by Slave, such as success, failure, etc.
* **HRESETn** : global reset signal, low effective, the reset can be directly connected to the system bus reset.
* **HCLK** , : Global clock signal.
* **HRDATA[31:0]** : Read data sent by Slave port.

OUTPUT信号：

* **HADDR[31:0]** : AHB bus address.
* **HWRITE** : сигнал индикатора чтения и записи, 1 - запись, 0 - считывание.
* **РАЗМЕР HSIZE[2:0]** : Используется для указания размера данных при каждой передаче.
* **HBURST[2:0]** : Используется для указания количества пакетных передач и метода увеличения адреса.
* **HPROT[3:0]** : сигнал управления защитой, используемый для указания типа данной передачи.
* **HTRANS[1:0]** : Используется для указания типа передачи. Всего существует четыре типа: ХОЛОСТОЙ ХОД / ЗАНЯТО / НЕПОСЛЕДОВАТЕЛЬНЫЙ / ПОСЛЕДОВАТЕЛЬНЫЙ, которые являются холостым ходом, занятостью, непоследовательной и непрерывной передачей.
* **HMASTERLOCK** : Заблокируйте основную передачу, мастер не может быть прерван в течение периода блокировки.
* **HWDATA[31:0]** : При записи данных максимальная поддержка составляет 1024 бита, рекомендуется от 32 до 256 бит.
* **HNONSEC** : Указывает, является ли текущая передача безопасной.
* **HEXCL** : Используется для указания, является ли передача эксклюзивной передачей (Эксклюзивный доступ).
* **HWSTRB** : Разрешена запись байтов данных, ширина бита равна ширине бита HWDATA, разделенной на 8, максимальное значение равно 128.

1.2 Подчиненный

Подчиненная сторона использует сигнал выбора HSELx, отправляемый декодером адреса, для ответа на передачу по шине.  
  
slave端输出信号:

* **HRDATA** : считываемые данные, максимальная разрядность составляет 1024 бита, рекомендуется от 32 до 256 бит.
* **ЧТЕНИЕ** : Сигнал готовности ведомого устройства, использование аналогично предварительному протоколу APB.
* **HRESP** : Ответный сигнал ведомого конечного устройства, указывающий, прошла ли текущая передача успешно, "1" неверен.
* **HEXOKAY** : Указывает, прошла ли эксклюзивная передача успешно.

译码器信号:

* **HSELx** : Сигнал выбора ведомого устройства, использование аналогично PSEL APB. Когда выбрано ведомое устройство и передача осуществляется без ХОЛОСТОГО хода, HSELx должен действовать одновременно с сигналами адреса и управления.

多路选择器信号:

* **HREADY** : Этот сигнал генерируется внутри шины для уведомления ведущего и ведомого устройств о том, что последняя передача завершена и шина готова к следующей передаче.

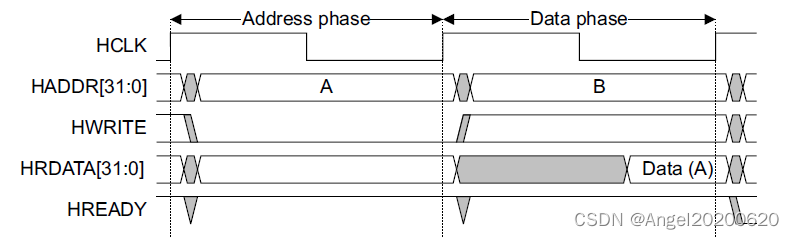
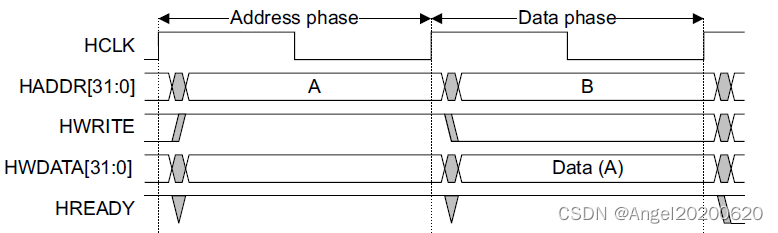
2 Передача

2.1 Базовая передача

AHB传输分两个阶段：地址阶段和数据阶段:

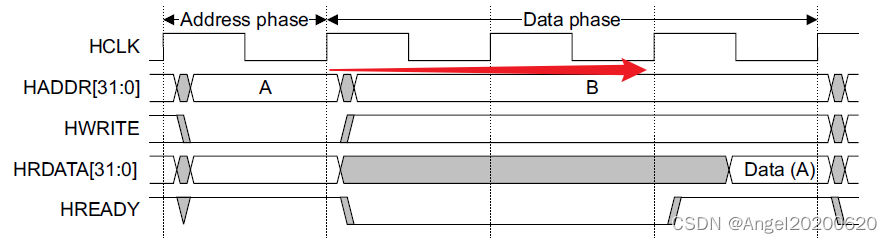
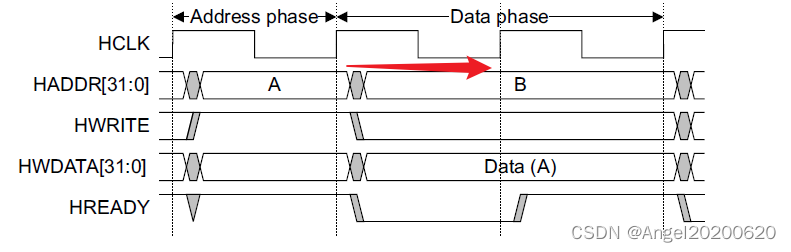
* Этап адресации: HWRITE, HADDR и другие управляющие сигналы действительны одновременно, а затем HREADY выводится на высокий уровень, чтобы указать, что текущий адрес вступил в силу.
* Этап передачи данных: HWDATA, HRDATA и другие сигналы, связанные с данными, действительны, указывая на начало чтения и записи данных. Этап передачи данных может занимать несколько тактов HCLK, в зависимости от размера передаваемых данных.

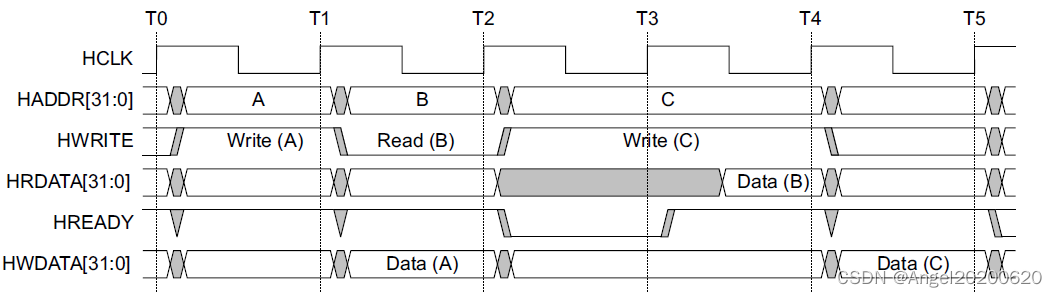
HWRITE信号控制数据传输方向:

1. HWRITE=1, означает передачу записи, мастер записывает данные в шину HWDATA[31:0].
2. HWRITE=0, означает передачу чтения, ведомое устройство отправляет данные на шину HRDATA[31:0].
3. На рисунке ниже показана однократная передача данных чтения и записи по шине AHB, включая немедленное чтение и немедленную запись.  
   **Передача чтения:**  
     
   **Передача записи:**  
   

Простая передача без состояний ожидания:

1. Следуя передовой технологии HCLK, мастер передает адресные и управляющие сигналы на шину.
2. Затем подчиненное устройство выполняет выборку адреса и управляющей информации на следующем восходящем фронте HCLK.
3. После того, как подчиненное устройство произвело выборку адресного и управляющего сигналов, оно может выдать ответ готовности. Выборка ответа производится на третьем восходящем фронте HCLK.

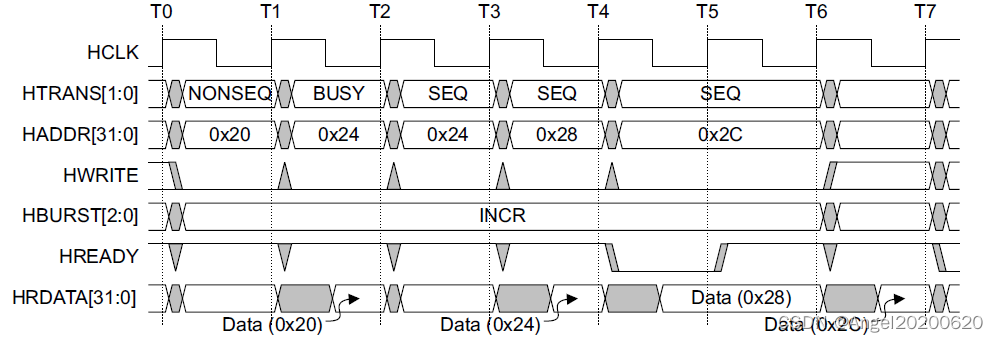
Этот простой пример демонстрирует, как фазы адреса и данных передачи происходят в течение разных тактов. Для любой передачи фаза адреса происходит перед фазой данных. Этот перекрывающийся адрес и данные являются основой для конвейерной обработки шины, обеспечивая высокопроизводительные операции и предоставляя подчиненным устройствам достаточное время для ответа на передачи.  
Подчиненные устройства могут вставлять состояния ожидания в любую передачу, чтобы получить дополнительное время завершения. Каждое подчиненное устройство имеет сигнал HREADYOUT, подаваемый на этапе передачи данных. Соединитель объединяет сигналы HREADYOUT от всех ведомых устройств для генерации управляющего сигнала HREADY, тем самым управляя всем процессом передачи.  
На следующем рисунке показана вставка ожидающих передач чтения и записи:  
**вставьте две ожидающие передачи чтения:**  
  
**вставьте один ожидающий передачи для записи:**  
  
Примечание: этап адреса и этап данных могут быть конвейерными, если можно использовать несколько последовательных передач, но с разными адресами, операция завершается в последовательности, показанной на рисунке ниже, и данные в A могут начать этап адреса B.



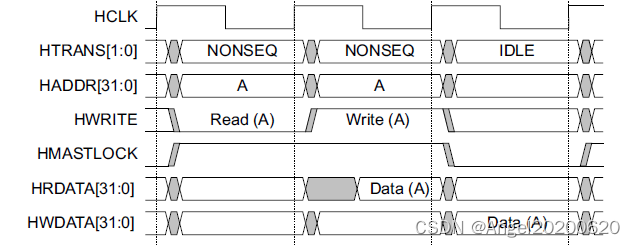
2.2 Тип передачи

HTRANS[1:0]:  
  
На рисунке ниже показаны типы передачи NONSEQ, BUSY и SEQ.

* T0-T1, выполните считывание с частотой 4 такта при передаче БЕЗ последовательности.
* T1-T2, ведущее устройство не может выполнить второй пакет и вставляет занятую передачу, чтобы задержать второй пакет, а подчиненная сторона возвращает данные считывания первого пакета.
* T2-T3, ведущее устройство готово инициировать вторую пакетную операцию, поэтому инициируется передача SEQ, и ведущее устройство игнорирует любые считанные данные, возвращенные подчиненным устройством.
* T3-T4, ведущая сторона выполняет третий пакет, а подчиненная сторона возвращает данные второго пакета.
* T4-T5, ведущее устройство выполняет последний пакет, а ведомое устройство не может завершить передачу, поэтому используйте HREADYOUT, чтобы вставить ожидание цикла.
* T5-T6, подчиненный конец возвращает считанные данные третьего пакета.
* T6-T7, подчиненный конец возвращает считанные данные последнего пакета.



3.3 Заблокированные передачи

Если ведущему устройству необходимо заблокировать доступ, оно также должно использовать сигнал HMASTLOCK. Этот сигнал указывает любому подчиненному устройству, что текущая последовательность передачи неделима и поэтому должна быть обработана перед обработкой других передач.  
Обычно заблокированные передачи используются для поддержания целостности сигнала, гарантируя, что ведомое устройство не выполняет никаких других операций между чтением и записью. Сигнал HMASTLOCK показан на рисунке ниже  
  
. Note: After a lock transfer, it is recommended that the master end insert an IDLE transfer.  
Most slaves do not need to implement the HMASTLOCK function, but only need to execute the instructions received in order. The slave can be accessed by multiple masters. For example, a Multi-Port Memory Controller (MPMC) must implement the HMASTLOCK signal.  
And at the beginning of IDLE data transmission, the master is also allowed to pull up the HMASTERLOCK, and pull down the HMASTLOCK signal in the non-IDLE state.

3.4 Transfer size

  HSIZE[2:0] represents the size of the data transfer, the following table lists the transfer size

| **РАЗМЕР HSIZE[2:0]** | **размер бит** | **Описание** |
| --- | --- | --- |
| 000 | 8 | Байт |
| 001 | 16 | Стать половиной |
| 010 | 32 | Word |
| 011 | 64 | Двойной мир |
| 100 | 128 | строка из 4 слов |
| 101 | 256 | строка из 8 слов |
| 110 | 512 | - |
| 111 | 1024 | - |

Примечание:  
Размер передачи определяется HSIZE, который должен быть меньше или равен разрядности данных шины. Например, для 32-разрядной ширины данных HSIZE должен использовать 0b000, 0b001 или 0b010. Используйте  
HSIZE объединен с HBURST для определения границы адреса.

3.5 Пакетная операция

Транспорт busrt поддерживает два типа: инкрементный и завершающий busrt.

* Увеличивающиеся пакеты получают доступ к последовательным местоположениям, и адрес каждой передачи в пакете является увеличивающимся адресом.
* Количество пакетов обертывания busrts контролируется HBURST и HSIZE . Например, для пакета с 4 битами адрес начала передачи равен 0x34, затем адресами передачи являются 0x34, 0x38, 0x3c и 0x30.
* Кодировка пакетного сигнала выглядит следующим образом:

| **HbURST[2:0]** | **Тип** | **Описание** |
| --- | --- | --- |
| 000 | ЕДИНЫЙ | одиночный пакет передачи |
| 001 | ВКЛ . | Увеличивающий пакет неопределенной длины |
| 010 | WRAP4 | Пакет обертывания с 4 битами |
| 011 | ИНКР4 | Пакет с увеличением частоты на 4 такта |
| 100 | WRAP8 | 8-битный пакет обертывания |
| 101 | INCR8 | 8-битный пакет с увеличением частоты |
| 110 | WRAP16 | 16-битный пакет обертывания |
| 111 | ИНКР16 | 16-битный пакет с увеличением частоты |

Примечание: Все пакетные передачи должны быть передачами с выравниванием по краю адреса, например, передачи слов должны быть выровнены по краю адреса слова HADDR[1:0] == 0b00.

**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

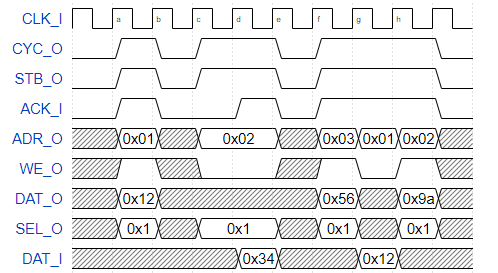


Рис. Пример временных диаграмм работы Wishbone Classic.

**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.

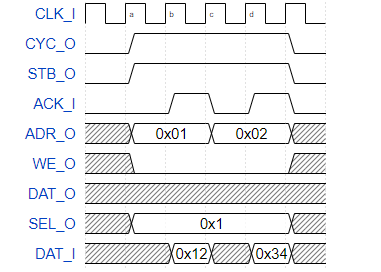


Рис. Временные диаграммы Wishbone Classic при работе с контроллером памяти.

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 сементика сигнала ACK\_I : CYC\_O=1 && STB\_O=1 && ACK\_I=1 ответ на запрос чтения.

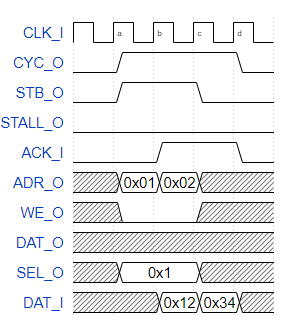


Рис. Временные диаграммы двух последовательных циклов чтения шины Wishbone Classic Pipelined bus.

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.

Литература

<https://habr.com/ru/articles/572926/>

<https://www.codetd.com/en/article/15908184>

<https://www.sobyte.net/post/2022-06/wishbone/>