8. SPI Core

8. Ядро SPI

Общее представление о ядре

SPI это индустриальный стандарт последовательного протокола, обычно используемый во встраиваемых системах для того, чтобы подключить микропроцессоры к различным внешним устройствам: сенсорам, конверторам, памяти и контроллерам. Ядро SPI с интерфейсом Avalon® реализует протокол SPI с одной стороны и предлагает интерфейс Avalon с распределением в памяти (Avalon-MM) с другой стороны.

Ядро SPI может реализовывать протокол либо мастер, либо слейв. Когда оно сконфигурировано как мастер, ядро SPI может контролировать до 32 независимый SPI слейвов. Ширина принимающего и передающего регистров может быть сконфигурирована между 1 и 32 битами. Большие размеры трансфертов поддерживаются программными процедурами. Ядро SPI предлагает выход запроса прерывания, который выставляет флаг по завершению трансферта.

Ядро SPI предназначено для SOPC Builder и легко интегрируется в любую систему, генерируемую SOPC Builder. Эта глава состоит из следующих секций:

- "Функциональное описание"
- "Программная модель" на стр. 8-8

Функциональное описание

Ядро SPI использует для связи две шины данных, шину контроля и синхросигнал:

- Мастер – выход, слейв – вход (mosi) – Выходные данные мастера поступают на входы слейвов;
- Мастер – вход, слейв – выход (miso) – Выходные данные слейва поступают на вход мастера;
- Последовательный такт (sclk) – тактовый сигнал, выдаваемый мастером слейву, используется для синхронизации битов данных;
- Выбор слейва (ss_n) – сигнал выбора (активный нуль), выдаваемый мастером конкретному слейв устройству, используется для целевого выбора устройства.

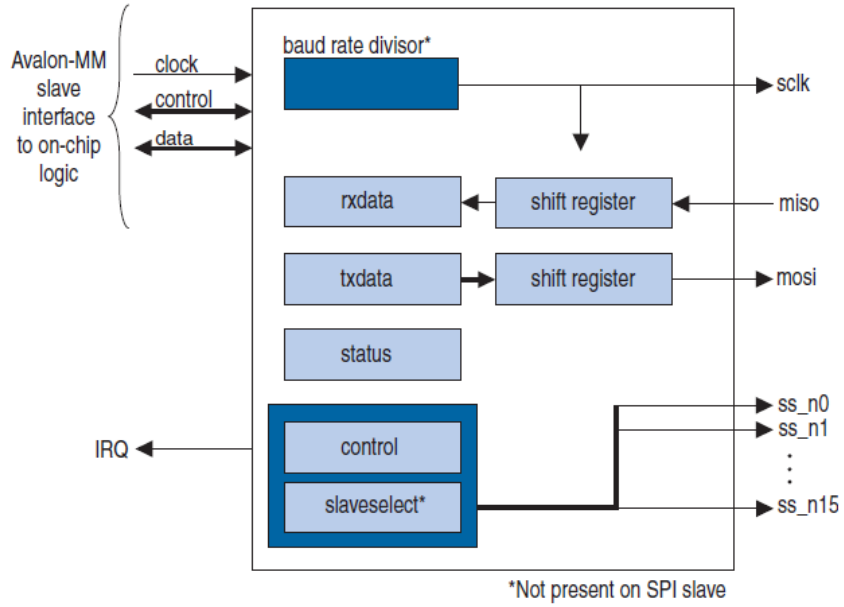
Ядро SPI имеет следующие видимые для пользователя средства:

- Регистровое пространство с распределением в памяти, состоящее из пяти регистров: rxdata, txdata, status, control и slaveselect
- Четыре интерфейсных порта SPI: sclk, ss_n, mosi и miso

Регистры предлагают интерфейс с ядром SPI и являются видимыми как слейв порт Avalon-MM. Порты sclk, ss_n, mosi и miso предлагают аппаратный интерфейс с другими устройствами SPI. Поведение sclk, ss_n, mosi и miso зависит от того, каким образом сконфигурировано ядро SPI: мастер или слейв.

На рис. 8-1 показана схема ядра SPI в режиме мастер.

Figure 8–1. SPI Core Block Diagram (Master Mode)



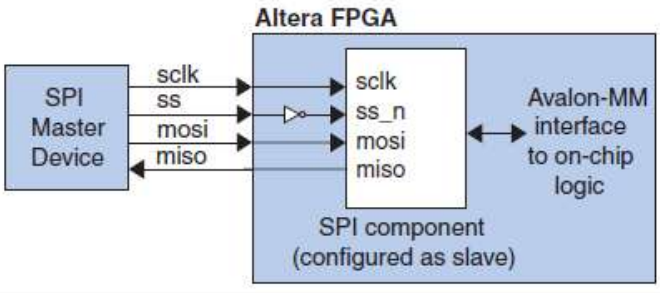
Логика ядра SPI синхронна с тактовым сигналом интерфейса Avalon-MM. Когда ядро сконфигурировано как мастер, оно делит тактовый сигнал Avalon-MM для генерирования выходного сигнала SCLK. Когда ядро сконфигурировано как слейв, логика приёмника ядра синхронизируется по входу SCLK. Интерфейс Avalon-MM ядра совместим с трансфертами Avalon-MM с управлением потоками. Ядро SPI может быть использовано совместно с контроллером DMA с управлением потоками для автоматизации трансферта данных между, например, ядром SPI и памятью.

За подробной информацией обратитесь к главе "Ядро интервального таймера".

Примеры конфигурации

На рис. 8-1 и 8-2 показаны две возможные конфигурации. На рис. 8-2 ядро SPI предлагает слейв интерфейс с внешним SPI мастером.

Figure 8–2. SPI Core Configured as a Slave



На рис. 8-1 ядро SPI предлагает мастер интерфейс, управляющий несколькими внешними слейв устройствами. Каждое слейв устройство на рис. 8-1 должно держать в тристабильном состоянии выход miso, пока его сигнал выбора не устрановлен.

Сигнал ss_n имеет активный нуль. Однако, любой сигнал может быть проинвертирован внутри FPGA, позволяя иметь сигналы выбора слейва либо активный нуль, либо активная единица.

Логика передатчика

Логика передатчика ядра SPI состоит из регистра удержания передатчика (txdata) и сдвигового регистра передатчика, каждый из которых шириной *n* бит. Ширина регистра *n* задаётся на стадии генерирования системы, и может быть целым числом от 8 до 32. После того, как мастер периферия записывает значение в регистр txdata, это значение копируется в сдвиговый регистр, чтобы потом передаться с началом следующей операции.

Сдвиговый регистр и регистр txdata обеспечивают двойную буферизацию во время передачи данных. Новое значение может быть записано в регистр txdata пока предыдущее значение сдвигается в сдвиговом регистре. Логика передатчика автоматически передаёт значение из регистра txdata в свободный сдвиговый регистр.

В мастер режиме передающий сдвиговый регистр прямо поступает на выход mosi. В слейв режиме передающий сдвиговый регистр прямо поступает на выход miso. Данные сдвигаются, начиная либо с младшего бита (LSB), либо со старшего бита (MSB), в зависимости от конфигурации ядра SPI.

Логика приёмника

Логика приёмника ядра SPI состоит из регистра удержания приёмника (rxdata) и сдвигового регистра приёмника, каждый из которых шириной *n* бит. Ширина регистра *n* задаётся на стадии генерирования системы, и может быть целым числом от 8 до 32. Мастер периферия принимает значение из регистра rxdata после того, как всё n-битное значение было получено в сдвиговом регистре.

Сдвиговый регистр и регистр rxdata обеспечивают двойную буферизацию во время приёма данных. Регистр rxdata может удерживать предыдущее значение принятых данных, пока новые данные последовательно не сдвигаются в сдвиговом регистре. Логика приёмника автоматически перемещает данные из сдвигового регистра в регистр rxdata по окончанию операции последовательного сдвига.

В мастер режиме данные приходят в сдвиговый регистр прямо со входа miso. В слейв режиме данные приходят в сдвиговый регистр прямо со входа mosi. Данные сдвигаются, начиная либо с младшего бита (LSB), либо со старшего бита (MSB), в зависимости от конфигурации ядра SPI.

Режимы мастер и слейв

На стадии генерирования системы разработчик может сконфигурировать ядро SPI либо в режиме мастер, либо в режиме слейв. Режимы не могут переключаться на стадии прогона.

Работа в режиме мастер

В режиме мастер, порты SPI имеют конфигурацию, показанную в табл. 8-1.

Табл. 8-1. Конфигурация в режиме мастер

Имя	Направление	Описание
mosi	выход	Выход данных в слейв устройства
miso	вход	Вход данных из слейв устройств
sclk	выход	Тактовый сигнал для всех слейвов
ss_nM	выход	Сигнал выбора слейва M, где M – число от 0 до 31

В режиме мастер разумный хост (например, микропроцессор) конфигурирует ядро SPI, используя регистры control и slaveselect, а затем записывает данные в буфер txdata для инициирования транзакции. Мастер периферия может контролировать процесс транзакции, читая регистр status. Мастер периферия может разрешить прерывания для уведомления хоста о поступивших новых данных (например, когда трансферт завершён), или о том, что буфер передатчика готов к приёму новых данных.

Протокол SPI имеет полный дуплекс, таким образом, за каждую транзакцию передаются и принимаются данные одновременно. Мастер передаёт новый бит данных на выход mosi, а слейв принимает новый бит данных на входе miso по каждому активному фронту sclk. Ядро SPI делит частоту системного такта Avalon-MM для генерирования сигнала sclk.

Когда ядро SPI сконфигурировано для интерфейса с несколькими слейвами, ядро имеет один сигнал ss_n для каждого слейва. Во время трансферта, мастер устанавливает ss_n для каждого слейва, заданного в регистре slaveselect. Обратите внимание, что не может быть более одного передатчика данных во время отдельного трансферта, иначе возникает конфликт на входе miso. Количество слейв устройств задаётся на стадии генерирования системы.

Работа в режиме слейв

В слейв режиме, порты SPI имеют конфигурацию, показанную в табл. 8-2.

Табл. 8-2. Конфигурация в режиме слейв

Имя	Направление	Описание
mosi	вход	Вход данных от мастер устройства
miso	выход	Выход данных в мастер устройство
sclk	вход	Тактовый сигнал
ss_nM	вход	Сигнал выбора

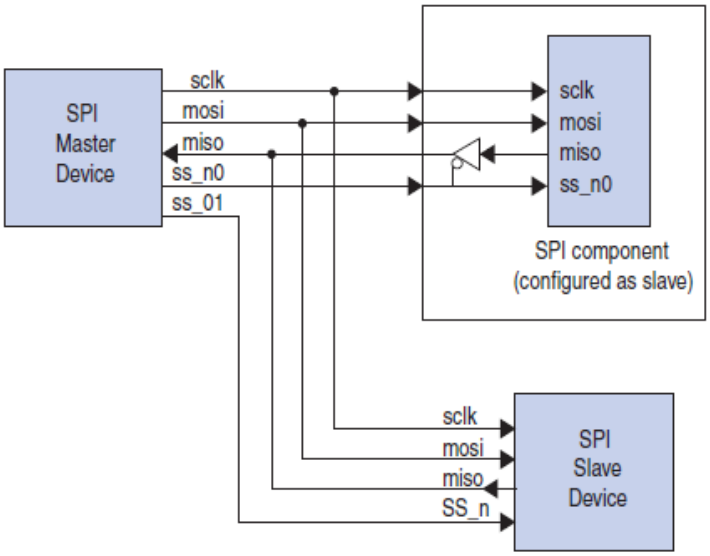
В режиме слейв, ядро SPI просто ожидает начала транзакции от мастера. Перед началом транзакции, логика слейва непрерывно опрашивает вход ss_n. Когда мастер устанавливает ss_n, слейв логика немедленно начинает посылать содержимое сдвигового регистра на выход miso. Слейв логика также получает данные на входе mosi, одновременно заполняя сдвиговый регистр приёмника. После того, как слово принято слейвом, мастер должен снять ss_n и заново установить этот сигнал, когда следующее слово будет готово к передаче.

Интеллектуальный хост, такой как микропроцессор, записывает данные в регистры txdata, чтобы они были переданы сразу, когда мастер инициирует передачу. Мастер периферия считывает принятые данные из регистра rxdata. Мастер периферия может разрешить прерывания, чтобы уведомлять хост, когда принимаются новые данные, или когда буфер передатчика готов для новых данных.

Среда с несколькими слейвами

Когда сигнал ss_n не установлен, ядра SPI устанавливают свои выходные выводы в высоко импедансное состояние. Предлагаемое Altera SPI слейв ядро переводит свои входы в неопределённое значение: либо 1, либо 0 – на своём выходе miso, когда оно не выбрано. Необходимы специальные ограничения, чтобы избежать конфликта сигналов на входе miso, если ядро SPI в режиме слейв подключено к внешнему мастер устройству SPI с несколькими слейвами. В таком случае, вход ss_n должен быть использован для контроля тристабильного буфера сигнала miso. На рис. 8-3 показан пример ядра SPI в режиме слейв в среде с двумя слейвами.

Figure 8–3. SPI Core in a Multi-Slave Environment



Интерфейс Avalon-MM

Интерфейс Avalon-MM ядра SPI состоит из одного слейв порта Avalon-MM. В дополнении к основным трансфертам слейв чтения и записи, ядро SPI поддерживает Avalon-MM трансферты чтения и записи с управлением потоками. Управление потоками запрещено, когда:

- включена опция disable flow control;
- выключена опция disable flow control, а мастер не поддерживает управление потоками.