Конфигурация

В этой секции описываются возможные опции конфигурации.

Настройка Master/Slave

Разработчик может выбрать либо мастер, либо слейв режим, чтобы определить поведение ядра SPI. Когда выбран режим мастер, доступны следующие опции: **Number of select (SS_n) signals, SPI clock rate** и **Specify delay**.

Опция Number of select (SS_n) signals

Этой настройкой задаётся количество слейвов, подключенных к SPI мастеру, в диапазоне от 1 до 32. SPI мастер имеет уникальный сигнал ss_n для каждого слейва.

Опция SPI Clock (sclk) Rate

Этой настройкой задаётся скорость обмена sclk - сигнала синхронизации между мастером и слейвом. Необходимая скорость может задаваться в Гц, кГц или МГц. Ядро мастер SPI использует системный тактовый сигнал Avalon-MM и делитель частоты для генерирования sclk.

Необходимая частота sclk может незначительно отличаться от предложенной. Доступные значения частоты являются:

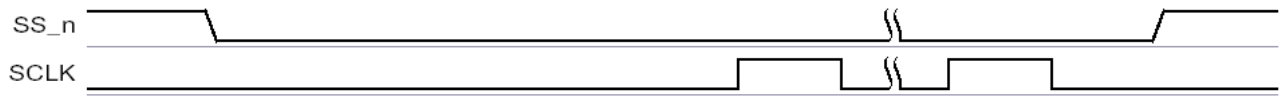
<*Avalon-MM системна тактовая частота*> / [2, 4, 6, 8, ...]

Полученное значение рабочей частоты не превышает значение необходимой частоты.

Опция Specify Delay

Включение этой опции вынуждает SPI мастер добавлять временную задержку между установкой сигнала ss_n и сдвигом первого бита данных. Эта задержка необходима большинству слейв устройств SPI. Если опция задержки включена, вы должны задать время задержки в нс, мкс или мс. Пример показан на рис. 8-4.

Figure 8–4. Time Delay Between Asserting ss_n and Toggling sclk



Логика генерирования задержки использует временное разбиение в полупериод от sclk. Полученная задержка является округлённым значением нескольких полупериодов сигнала sclk, как это показано в формулах 8-1 и 8-2.

Equation 8–1.

$$p = \frac{1}{2} \times (\text{period of sclk})$$

Equation 8–2.

$$\text{actual delay} = \text{ceiling} \times \left(\frac{\text{desired delay}}{p} \right) \times p$$

Настройка Data Register

Настройки регистра данных влияют на размер и поведение регистра данных ядра SPI. Существуют две настройки:

- **Width** – Эта настройка задаёт ширину регистров rxdata, txdata, а также сдвиговых регистров приёмника и передатчика. Диапазон от 1 до 32.
- **Shift direction** – Эта настройка определяет направление сдвига во входном и в выходном регистрах (сначала старший бит (MSB) или сначала младший бит (LSB)).

Настройка Timing

Временные настройки влияют на взаимодействие между сигналами ss_n, sclk, mosi и miso. Здесь сигналы mosi и miso упоминаются как *data*. Есть две временные настройки:

- **Clock polarity** – Эта настройка может быть 0 или 1. Когда полярность такта 0 – состояние ожидания sclk является 0. Когда полярность – 1, состояние ожидания sclk – 1.
- **Clock phase** - Эта настройка может быть 0 или 1. Когда фаза тактового сигнала 0, данные защёлкиваются по переднему фронту sclk, а сменяются по заднему фронту. Когда фаза тактового сигнала 1, данные защёлкиваются по заднему фронту sclk, а сменяются по переднему фронту.

На рис. 8-5 – 8-8 показано поведение сигнала во всех возможных случаях полярности и фазы тактового сигнала.

Figure 8–5. Clock Polarity = 0, Clock Phase = 0

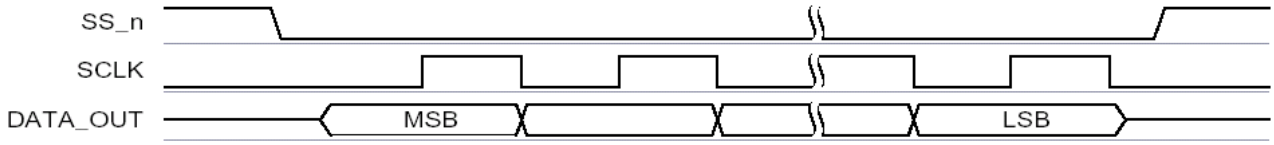


Figure 8–6. Clock Polarity = 0, Clock Phase = 1

Figure 8–7. Clock Polarity = 1, Clock Phase = 0

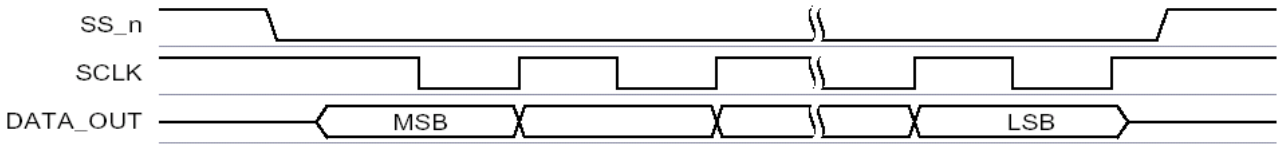
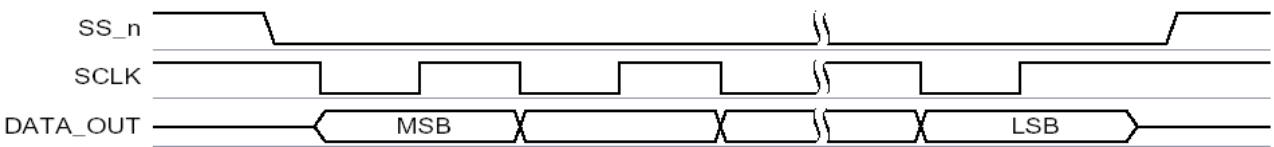


Figure 8–8. Clock Polarity = 1, Clock Phase = 1



Программная модель

В следующих секциях описывается программная модель для ядра SPI, включая карту регистров и программные конструкции, используемые для доступа к аппаратной части. Для пользователей процессора Nios II, Altera предлагает заголовочный файл системной библиотеки HAL, в котором определены регистры ядра SPI. Ядро SPI не является какой-либо категорией групповых моделей устройств, поддерживаемой HAL, поэтому к нему нельзя обращаться через HAL API или стандартную библиотеку ANSI Си. Altera предлагает специальную процедуру доступа к аппаратной части ядра SPI.

Процедура аппаратного доступа

Altera предлагает одну процедуру доступа, alt_avalon_spi_command(), которая предлагает основной доступ к ядру SPI, сконфигурированному в качестве мастера.

alt_avalon_spi_command()

Прототип:
```
int alt_avalon_spi_command(alt_u32 base, alt_u32 slave,
                           alt_u32 write_length,
                           const alt_u8* wdata,
                           alt_u32 read_length,
                           alt_u8* read_data,
                           alt_u32 flags)
```

Многопоточность (Thread-safe): Нет.

Доступность из программы обработки прерываний (ISR): Нет.

Включения: <altera_avalon_spi.h>

Описание: Эта функция выполняет последовательный контроль по шине SPI. Она поддерживает только SPI мастер с шириной данных меньше или равных 8 бит. Один вызов этой функции записывает буфер данных произвольной длины в порт mosi, а затем читает обратно произвольную длину данных из порта miso. Функция выполняет следующие действия:

1. Устанавливает выход выбора слейва для заданного слейв устройства. Выход первого слейв устройства – 0.
2. Передаёт write_length байты данных из wdata по интерфейсу SPI, отбраковывая входящие данные по порту miso.
3. Читает read_length байты данных и сохраняет их в буфере, перенаправленном с read_data. Порт mosi установлен в нуль во время транзакции чтения.
4. Снимает назначение с выхода выбора слейва, за исключением ситуации, когда поле флага содержит значение ALT_AVALON_SPI_COMMAND_MERGE. Если вы хотите передавать из различных буферов, вызывайте функцию несколько раз и задайте флаг объединения для всех событий доступа, за исключением последнего. Для доступа к шине SPI, состоящей из нескольких потоков, вы должны использовать семафор или флаг, чтобы следить за тем, что только один поток исполняет функцию в данное время.

Возвращает: число байтов, хранящихся в буфере read_data.

Программные файлы

Ядро SPI связано со следующими программными файлами. Эти файлы предлагают аппаратный интерфейс с устройством.

- **altera_avalon_spi.h** – Этот файл определяет карту регистров, предлагает символьные константы для доступа на аппаратном уровне.

- **altera_avalon_spi.c** – Этот файл реализует низкоуровневые процедуры доступа к устройству.

Карта регистра

Мастер периферия на Avalon-MM контролирует и связывается с ядром SPI посредством шести 32-битных регистров, показанных в таблице 8-3. Таблица предполагает n-битную ширину данных для rxdata и txdata.

Табл. 8-3. Карта регистра мастер устройства SPI

Внутр. адрес	Имя регистра	Тип [R/W]	32..11	10	9	8	7	6	5	4	3	2	1	0
0	rxdata*(1)*	R	RXDATA (n-1..0)											
1	txdata*(1)*	W	TXDATA (n-1..0)											
2	status*(2)*	R/W				E	RRDY	TRDY	TMT	TOE	ROE			
3	control	R/W		SSO*(3)*		IE	IRRDY	ITRDY		ITOE	IROE			
4	Reserved	-												
5	slaveselect*(3)*	R/W	Slave Select Mask											

Примечания к табл. 8-3:
(1) Биты с 31 по n имеют неопределённое значение, когда n меньше 32.
(2) Операция записи в регистр status сбрасывает биты ROE, TOE и E.
(3) Существует только в режиме мастер.

Чтение неопределённых битов возвращает неопределённое значение. Запись в неопределённые биты не имеет эффекта.

Регистр rxdata

Мастер периферия читает принимаемые данные из регистра rxdata. Когда сдвиговый регистр приёмника принимает все n бит данных, бит RRDY регистра status устанавливается в 1 и данные передаются в регистр rxdata. Чтение регистра rxdata сбрасывает бит RRDY. Запись в регистр rxdata не имеет эффекта.

Новые данные всегда передаются в регистр rxdata, не смотря на то, что предыдущие данные остались не востребованы. Если RRDY – 1, то данные передаются в регистр rxdata (не смотря на то, что предыдущие данные не востребованы), возникает ошибка переполнения приёмника, а бит ROE регистра status устанавливается в 1. В этом случае содержимое регистра rxdata имеет неопределённое значение.

Регистр txdata

Мастер периферия записывает данные, предназначенные для передачи, в регистр txdata. Когда бит TRDY регистра status установлен в 1, это означает, что регистр txdata готов к получению новых данных. Бит TRDY находится в 0, пока происходит запись в регистр txdata. Бит TRDY устанавливается в 1 после того, как данные передаются в сдвиговый регистр передатчика, означая, что регистр удержания txdata готов к приёму новых данных.

Мастер периферия не сможет записывать в регистр txdata пока передатчик готов получить новые данные. Если TRDY – 0 и мастер периферия пишет новые данные в регистр txdata, возникает ошибка переполнения передатчика, а бит TOE регистра status устанавливается в 1. В этом случае новые данные игнорируются, и содержимое регистра txdata не может быть изменено.

Например, предположим, что ядро SPI готово к работе (это означает, что регистр txdata и сдвиговый регистр передатчика - пустые), тогда CPU записывает значение данных в регистр удержания txdata. Бит TRDY моментально устанавливается в 0, но, когда данные из регистра txdata перемещаются в сдвиговый регистр передатчика, TRDY возвращается в 1. CPU записывает второе значение данных в регистр txdata, и снова бит TRDY устанавливается в 0. В это время сдвиговый регистр находится в состоянии последовательной передачи данных, таким образом, бит TRDY остаётся в нуле, пока не завершится операция сдвига. Когда операция завершена, второе значение данных передаётся в сдвиговый регистр передатчика, а TRDY снова устанавливается в 1.

Регистр status

Регистр status состоит из битов, отображающих статус состояния ядра SPI. Каждый бит ассоциирован с соответствующим битом разрешения прерывания в регистре control, это обсуждается в секции "Регистр control" на стр. 8-11. Мастер периферия может читать регистр status в любое время, не изменяя значений битов. Запись в регистр status сбрасывает биты ROE, TOE и E. В табл. 8-4 описаны индивидуальные биты регистра status.

Табл. 8-4. Биты регистра status

№	Имя	Описание
3	ROE	**Ошибка переполнения приёмника** Бит ROE устанавливается в 1, когда новые данные были приняты при полном регистре rxdata (т.е. бит RRDY в 1). В этом случае новые данные замещают старые. Запись в регистр status сбрасывает бит ROE в 0.
4	TOE	**Ошибка переполнения передатчика** Бит TOE устанавливается в 1, когда новые данные были в полный регистр txdata (т.е. бит TRDY в 0). В этом случае новые игнорируются. Запись в регистр status сбрасывает бит TOE в 0.
5	TMT	**Сдвиговый регистр передатчика пуст** В режиме мастер, бит TMT устанавливается в 0, когда транзакция находится в процессе, и устанавливается в 1, когда сдвиговый регистр пуст. В режиме слейв, бит TMT устанавливается в 0, когда слейв выбран (SS_n в нуле), или когда SPI слейв интерфейсный регистр не готов к приёму данных.
6	TRDY	**Передатчик готов** Бит TRDY устанавливается в 1, когда регистр txdata пуст.
7	RRDY	**Приёмник готов** Бит RRDY устанавливается в 1, когда регистр rxdata заполнен.
8	E	**Ошибка** Бит E является логическим ИЛИ битов TOE и ROE. Это механизм детектирования состояния ошибки для программиста. Запись в регистр status сбрасывает бит E в 0.

Регистр control

Регистр control состоит из битов данных, контролирующих работу ядра SPI. Мастер периферия может читать регистр control в любое время, не изменяя значение никакого бита.

Большинство битов (IROE, ITOE, ITRDY, IRRDY и IE) регистра контролируют прерывания для состояний статуса, описанных в регистре status. Например, бит 1 регистре status - это ROE (ошибка переполнения приёмника), а бит 1 регистра control - это IROE, который разрешает прерывание для состояния ROE. Ядро SPI посылает запрос прерывания, когда оба соответствующих бита в регистрах status и control в 1.

Биты регистра control показаны в табл. 8-5.

Табл. 8-5. Биты регистра control

№	Имя	Описание
3	IROE	Установка IROE в 1 разрешает прерывания для ошибки переполнения приёмника
4	ITOE	Установка ITOE в 1 разрешает прерывания для ошибки переполнения передатчика
6	ITRDY	Установка ITRDY в 1 разрешает прерывания для состояния готовности передатчика
7	IRRDY	Установка IRRDY в 1 разрешает прерывания для состояния готовности приёмника
8	IE	Установка IE в 1 разрешает прерывания для любого состояния ошибки
10	SSO	Установка SSO в 1 принуждает ядро SPI управлять собственными выходами ss_n, независимо от того, происходил ли операция сдвига или нет. Регистр slaveselect контролирует то, как назначать выводы ss_n. SSO может быть использован для передачи или приёма данных произвольного размера, например, более 32 бит.

После сброса все биты регистра control устанавливаются в 0. Все прерывания запрещены, а сигналы ss_n не назначены.

Регистр slaveselect

Регистр slaveselect является битовой маской для сигналов ss_n управляемых SPI мастером. Во время операции последовательного сдвига, SPI мастер выбирает только слейв устройства, заданные в регистре slaveselect.

Регистр slaveselect существует только в SPI ядре, сконфигурированном в качестве мастера. Один бит регистра slaveselect соответствует конкретному выходу ss_n, что определяется разработчиком на стадии генерирования системы.

Мастер периферия может выбрать одновременно несколько бит slaveselect, принуждая мастер SPI выбрать одновременно несколько слейв устройств для выполнения транзакций. Например, чтобы разрешить связь со слейв устройствами 1, 5 и 6, необходимо выбрать 1, 5 и 6 биты регистра slaveselect. Однако необходимо избегать конфликтов сигнала между несколькими слейв устройствами по их выходу miso.

После сброса, бит 0 устанавливается в 1, а все остальные биты сбрасываются в 0. Т.е. после сброса чипа, автоматически выбирается устройство 0.