Языки описания схем

(mk.cs.msu.ru → Лекционные курсы → Языки описания схем)

Блок K4

Кое-что ещё:
SPI для двух устройств

Лектор:
Подымов Владислав Васильевич

E-mail:
valdus@yandex.ru

ВМК МГУ, 2019, осенний семестр

База SPI: общее описание

SPI (Serial Peripheral Interface) — это самое простое семейство протоколов синхронной последовательной передачи данных между устройствами

Это семейство дуплексных протоколов: данные пересылаются одновременно в две стороны

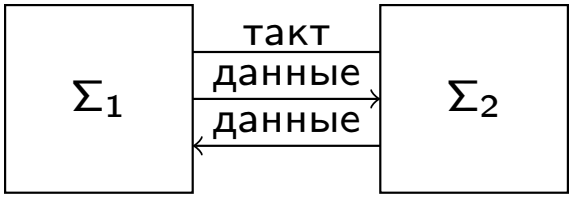
Начнём с самого простого (базового) варианта этого протокола: попробуем организовать синхронную передачу чисел ширины 8 между **двумя** устройствами

Дуплексность: для пересылки данных потребуются два провода

Синхронность: для пересылки часов потребуется *хотя бы* один провод

Остановимся на **трёх** проводах

База SPI: общее описание



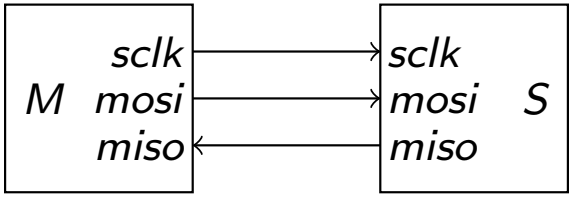
Кто кому должен переслать тактовый сигнал?

Заранее договоримся о том, какое из устройств является ведущим (master; M), а какое — ведомым (slave; S)

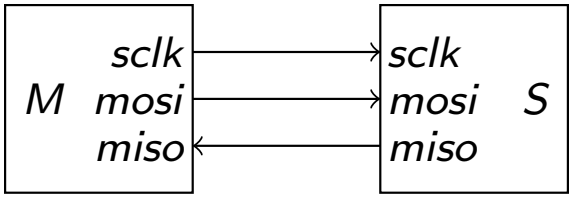
Ведущее устройство пересылает тактовый сигнал ведомому, и оба устройства *сэмплируют* принимаемые данные по фронтам этого сигнала

Общепринятые названия портов (на картинке — сверху вниз):

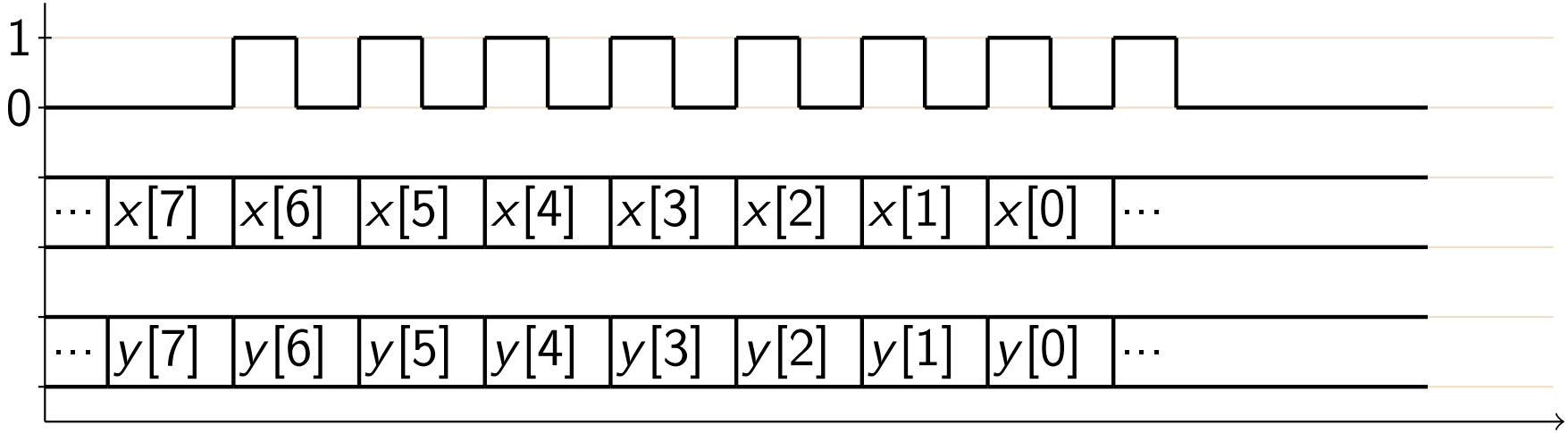
- sclk (serial clock)
- mosi (master out slave in; выход ведущего, вход ведомого)
- miso (master in slave out; выход ведущего, вход ведомого)



База SPI: общее описание



Сразу к делу — пересылка одной пары сообщений ширины 8 (x по *mosi*, y по *miso*) в одном из вариантов протокола выглядит так:



Ведущая схема решает, проводить ли передачу

Передачи нет ⇒ тактовый сигнал не осциллирует (тишина)

Передача есть ⇒ тактовый сигнал осциллирует 8 раз, по каждому главному фронту (здесь — переднему) запоминается значение очередного бита данных, и после фронта выводится следующее значение

База SPI: общее описание

Другие варианты передачи между двумя устройствами

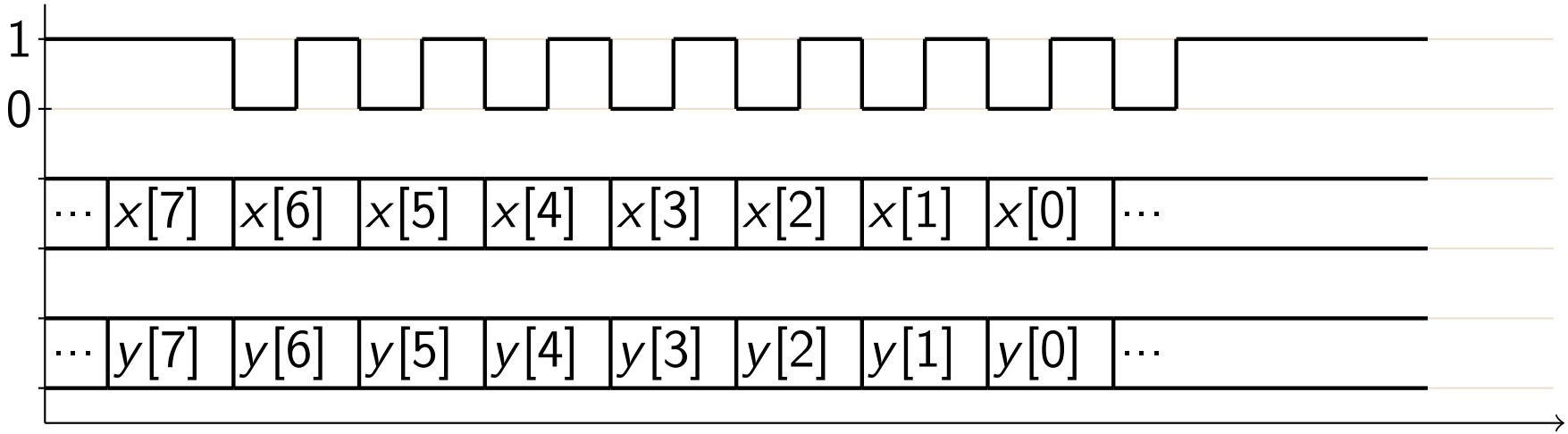
Другая ширина сообщения ⇒ соответствующее количество осцилляций (*но число битов должно быть оговорено заранее*)

- *Самая популярная ширина* — 8

Другой порядок пересылки — изменения в протоколе очевидны

- *Самый популярный порядок — от старшего бита к младшему*

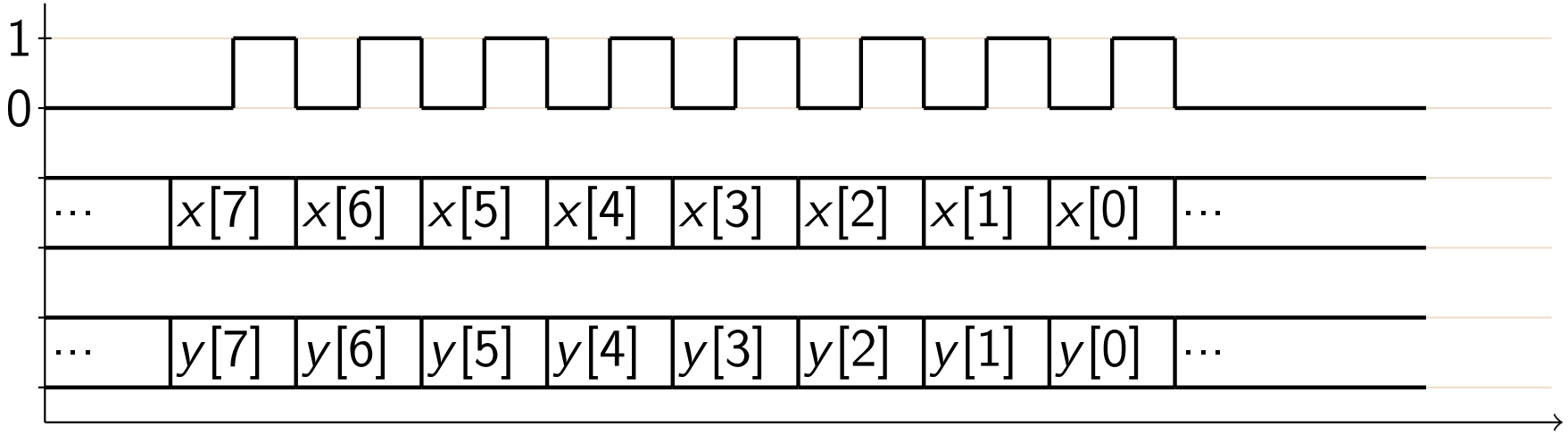
Задний фронт в качестве главного:



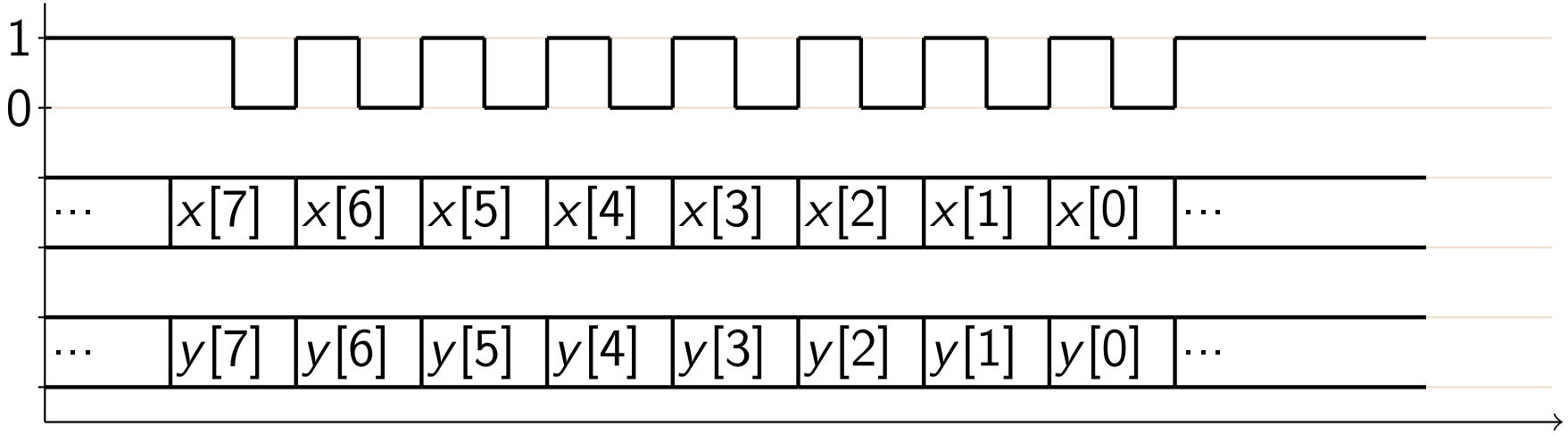
База SPI: общее описание

Другие варианты передачи между двумя устройствами

Смещение данных на половину такта



<...> и задний фронт в качестве главного:



База SPI: реализация приёмника

В реализации UART приёмник был устроен заметно сложнее передатчика:

- Передатчик пересылает биты сообщения согласно протоколу, не думая о погрешностях и том, как сообщение будет приниматься
- Приёмник должен корректно принять сообщение для любых частоты, фазы и (*разумных*) погрешностей передатчика, отталкиваясь только от частоты протокола

Сложность приёмника по сравнению с передатчиком — это одна из характерных черт *асинхронных* протоколов

В простом базовом варианте SPI (*и других синхронных протоколов*) подсхему, принимающую сообщение, реализовать проще, чем передающую

Далее будем *полагать*, что оба устройства знают время начала и конца передачи каждого сообщения

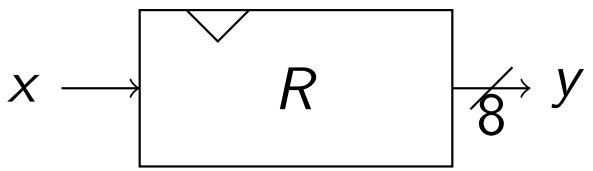
База SPI: реализация приёмника

Всё, что нужно уметь делать при приёме сообщения, —

- принимать очередной бит по главному фронту тактового сигнала протокола и
- выдавать последние 8 принятых битов

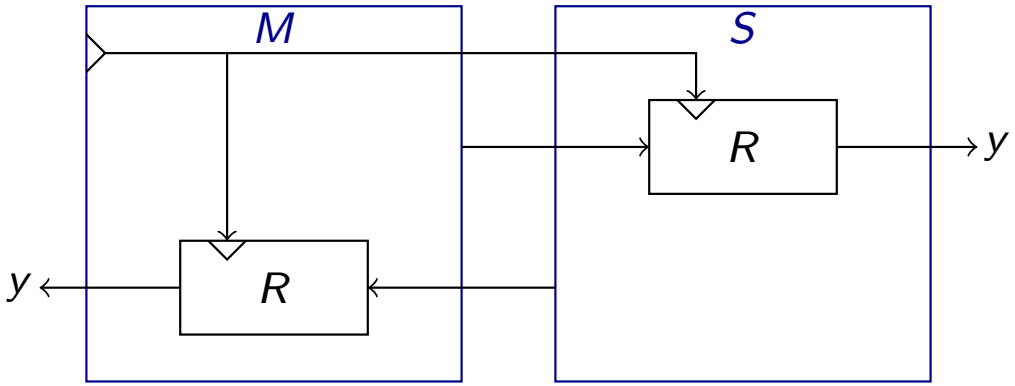
Это умеет делать с *сдвиговый регистр* R ширины 8:

- при чтении очередного бита b со входа x текущее состояние $(q_7 q_6 \ldots q_0)$ заменяется на $(q_6 q_5 \ldots q_0 b)$
- текущее состояние направляется на выход y



База SPI: реализация приёмника

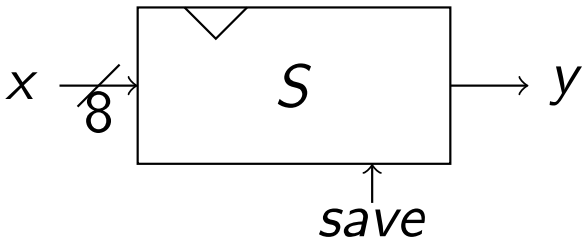
Устройство SPI-приёмников, направляющих принятые сообщения в соответствующие выходы y:



База SPI: реализация передатчика

SPI-передатчик можно устроить так:

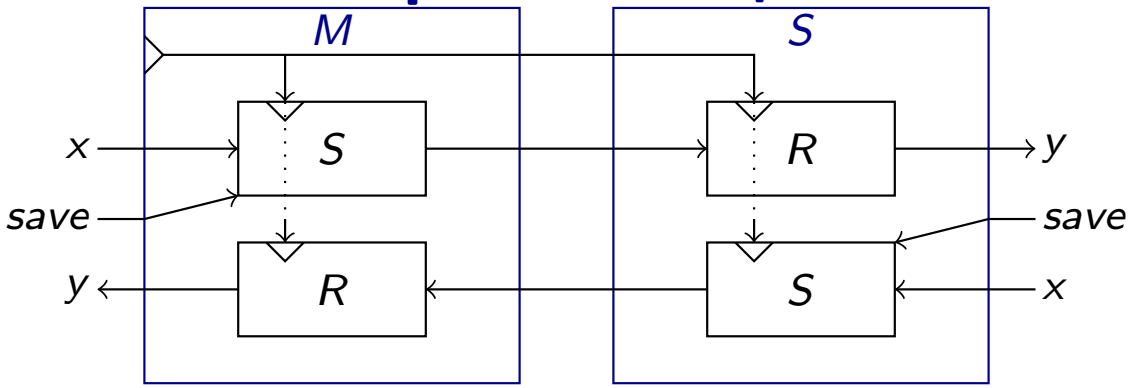
- согласно управляющему сигналу на входе (*save*) он сохраняет сообщение (x) и направляет на выход (y) старший разряд этого сообщения
- по каждому главному фронту тактового сигнала протокола он переходит к выдаче следующего разряда



Схемы с похожим поведением уже возникали при обсуждении протокола *UART* и до этого обсуждения: это *сериализаторы*

Сериализатор — типичный элемент схемной реализации *асинхронных* протоколов

База SPI: итоговая реализация

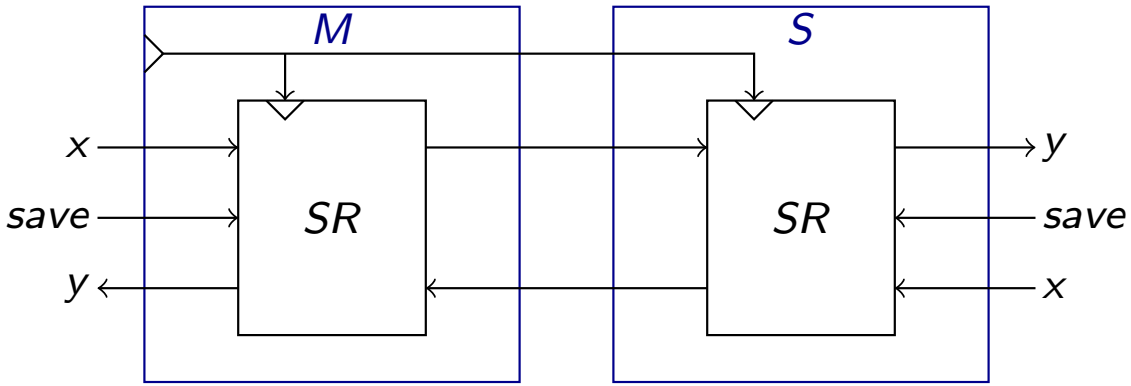


Подсхемы S и R имеют схожее поведение, и в связи с этим их можно объединить в единую подсхему SR — синхронный регистр, умеющий

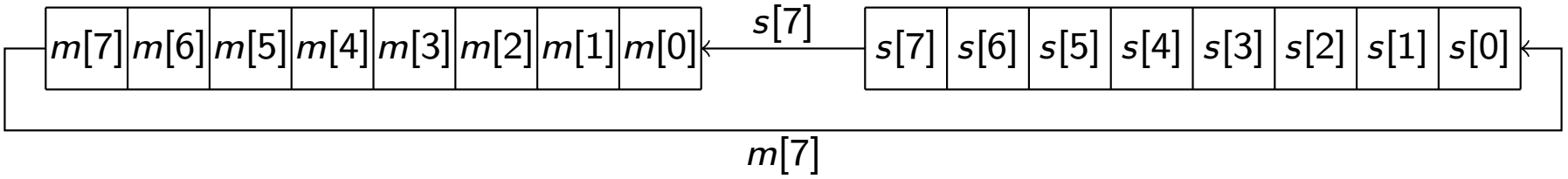
- хранить число ширины 8
- выдавать хранящееся число и его старший разряд
- сохранять число со входной шины (*как это делает параллельный регистр*)
- сдвигать число в сторону старшего бита, заполняя младший бит значением на одноразрядном входе (*как это делает сдвиговый регистр*)

После такого объединения при пересылке сообщения в SR одновременно хранятся принятые биты и биты, которые осталось отправить

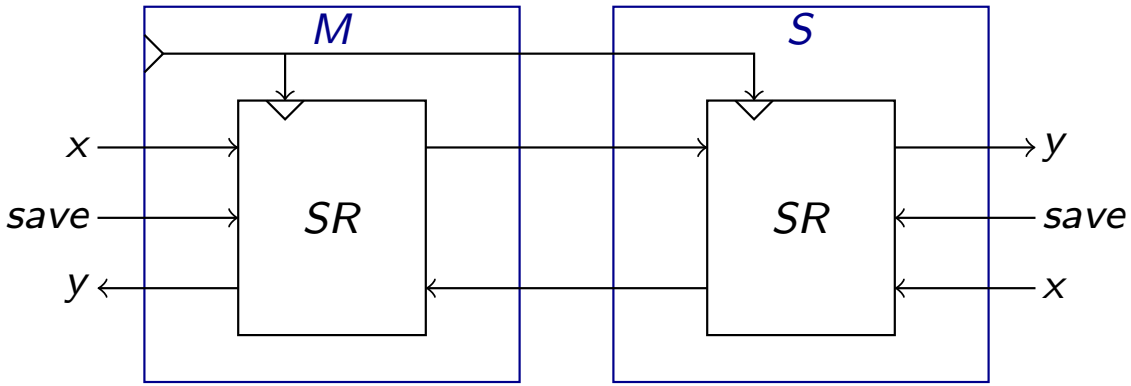
База SPI: итоговая реализация



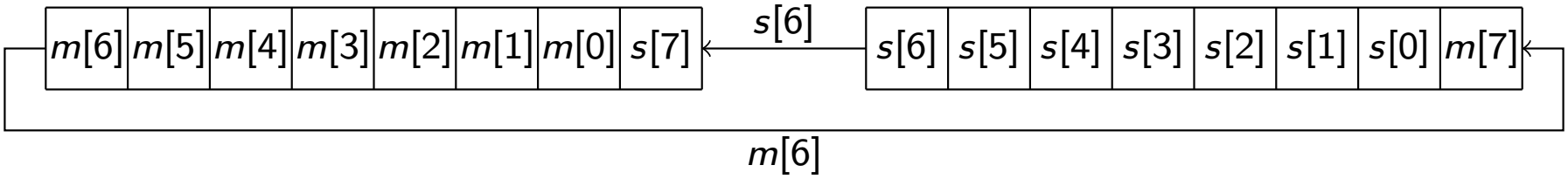
Пересылка одной пары сообщений для “объединённого” регистра *SR* выглядит так:



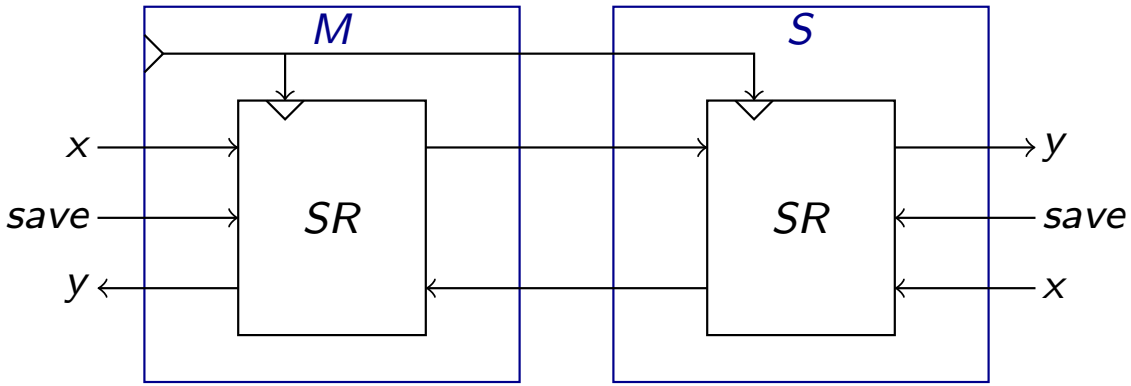
База SPI: итоговая реализация



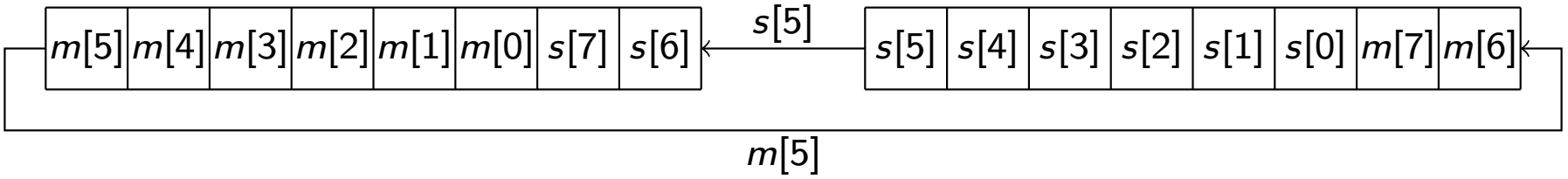
Пересылка одной пары сообщений для “объединённого” регистра *SR* выглядит так:



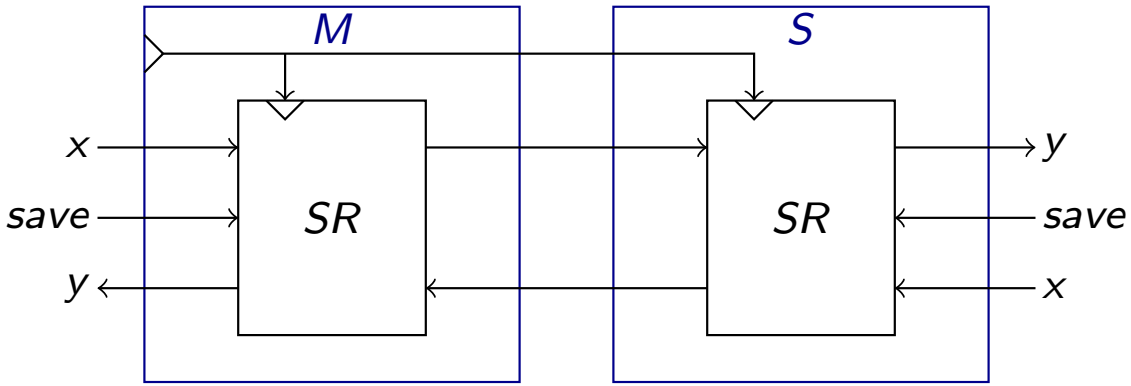
База SPI: итоговая реализация



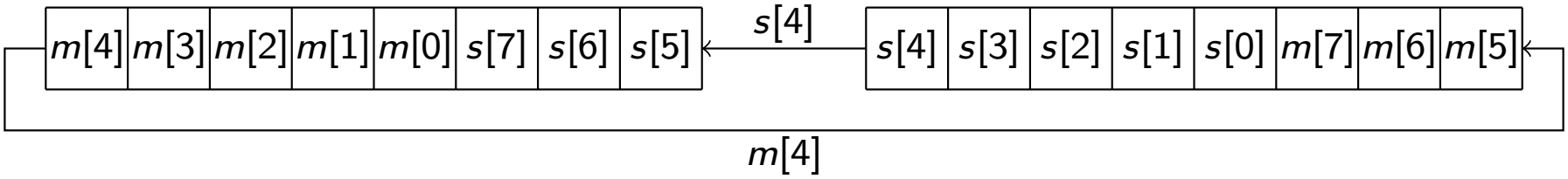
Пересылка одной пары сообщений для “объединённого” регистра *SR* выглядит так:



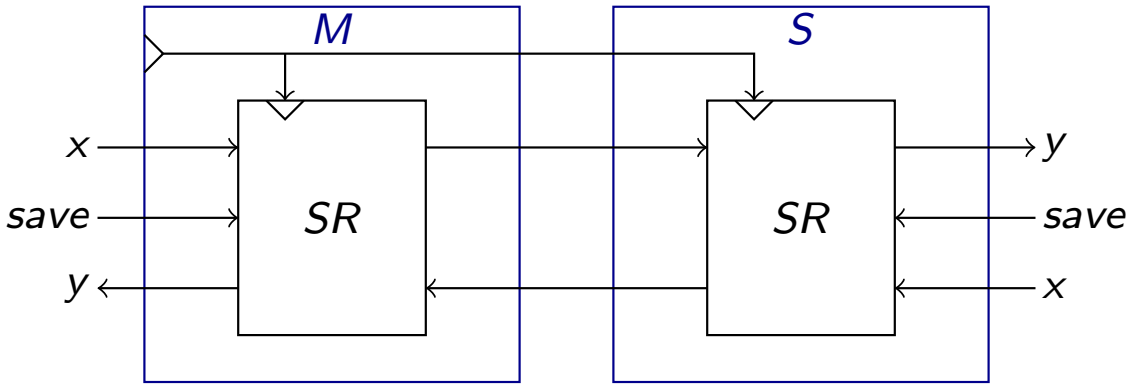
База SPI: итоговая реализация



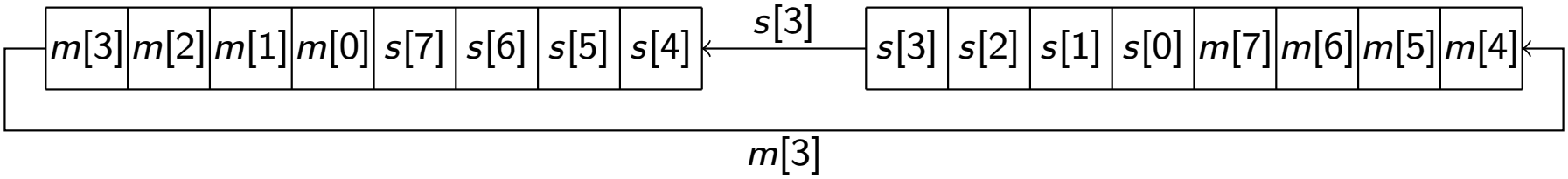
Пересылка одной пары сообщений для “объединённого” регистра *SR* выглядит так:



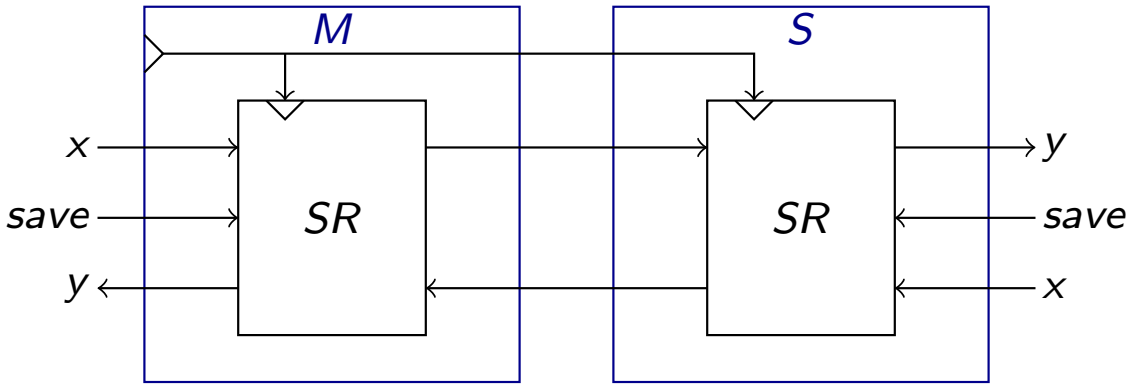
База SPI: итоговая реализация



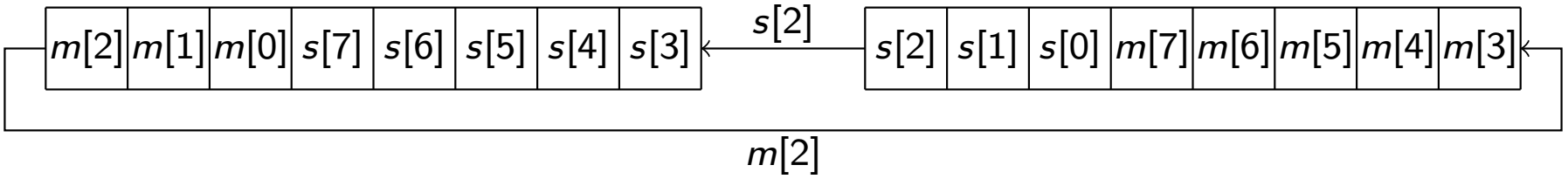
Пересылка одной пары сообщений для “объединённого” регистра *SR* выглядит так:



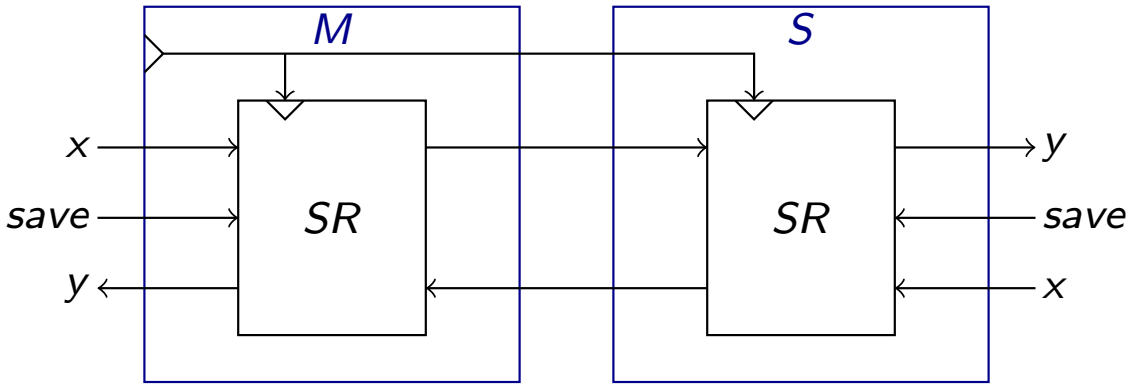
База SPI: итоговая реализация



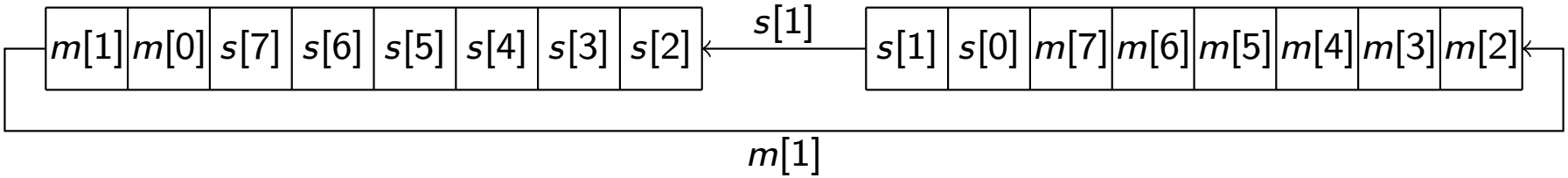
Пересылка одной пары сообщений для “объединённого” регистра *SR* выглядит так:



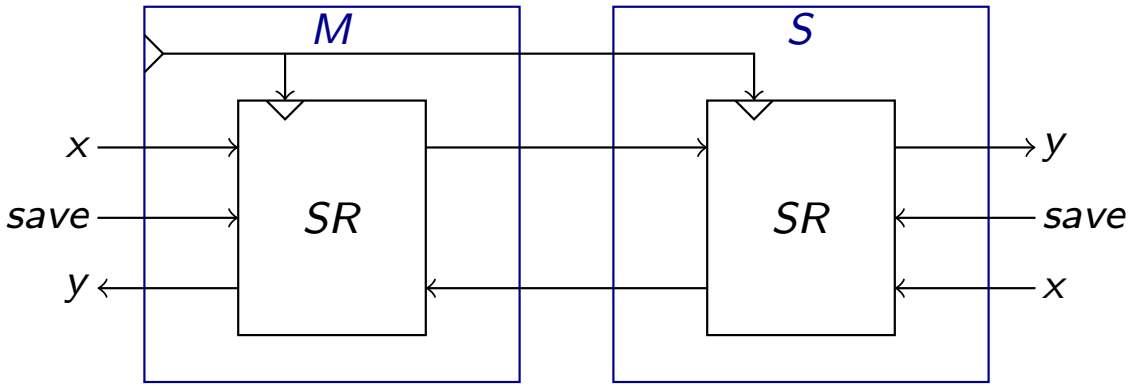
База SPI: итоговая реализация



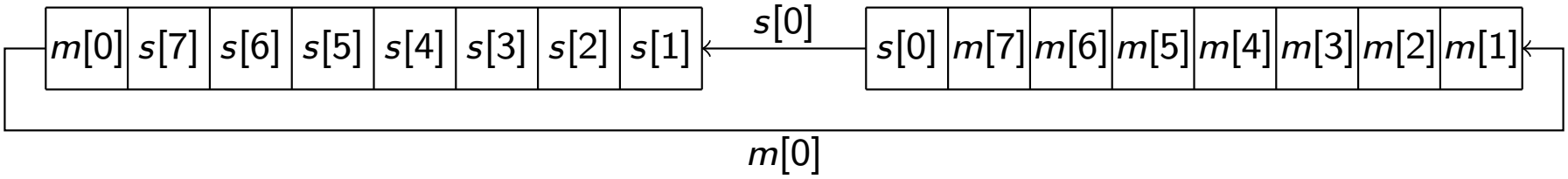
Пересылка одной пары сообщений для “объединённого” регистра *SR* выглядит так:



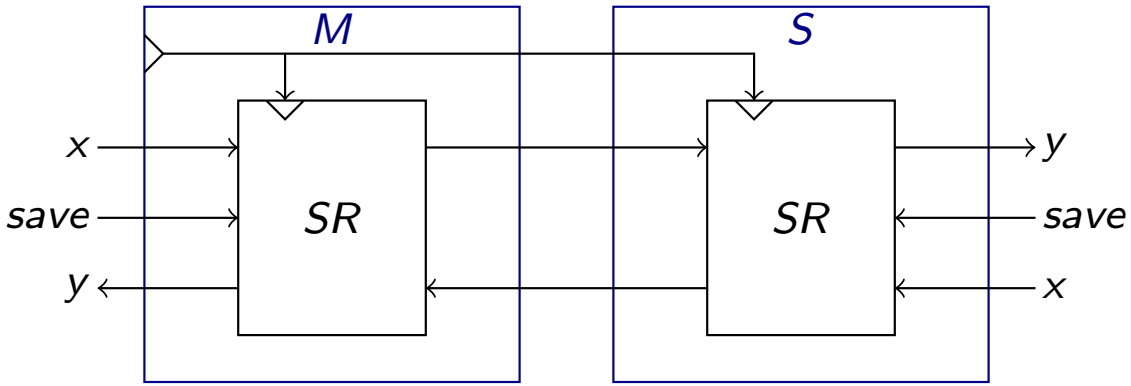
База SPI: итоговая реализация



Пересылка одной пары сообщений для “объединённого” регистра *SR* выглядит так:



База SPI: итоговая реализация



Пересылка одной пары сообщений для “объединённого” регистра *SR* выглядит так: