Языки описания схем

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

Блок K5

Кое-что ещё:
SPI для произвольного числа устройств
Состояние высокого импеданса

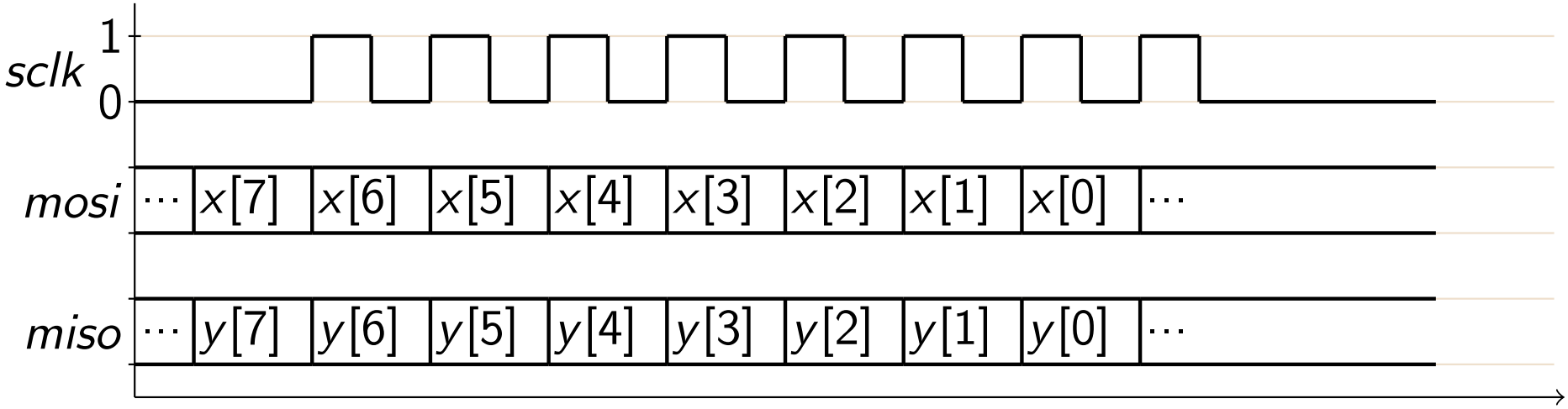
Лектор:
Подымов Владислав Васильевич

E-mail:
valdus@yandex.ru

ВМК МГУ, 2019, осенний семестр

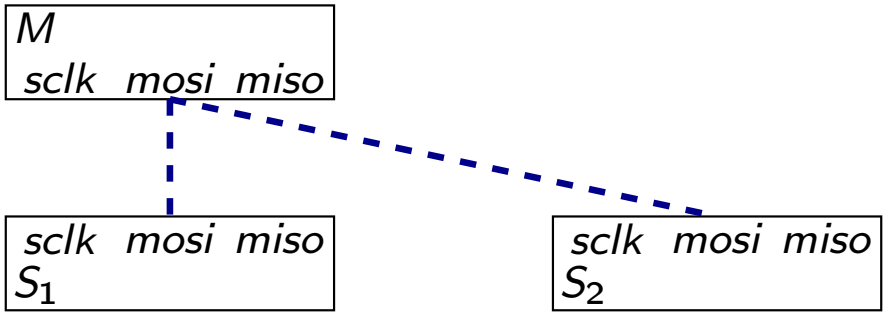
SPI: общее описание

Пересылка пары сообщений по протоколу SPI:



Протокол SPI позволяет организовать обмен данными между центральным устройством (ведущим) и **произвольным числом** периферийных устройств (ведомых)

Для ясности представим себе, что хочется передавать данные между ведущим устройством (M) и **двумя** ведомыми (S_1, S_2):

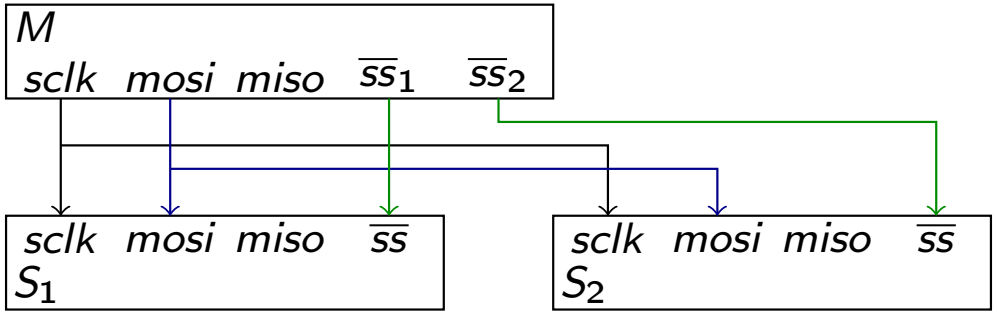


SPI: sclk, ss, mosi

Передать тактовый сигнал от M к S_1 и S_2 — это просто

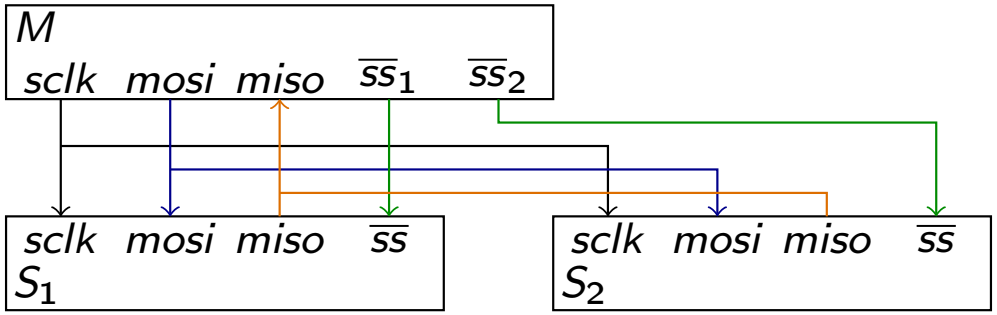
Передать данные от M к ведомым устройствам — это чуть сложнее, но тоже достаточно просто:

- Направим данные от M на соответствующий вход **каждого** ведомого устройства
- Соединим M с каждым ведомым устройством дополнительным проводом
  - ss = slave select, сигнал выбора ведомого устройства: $1 \Rightarrow M$ обменивается данные с соответствующим ведомым устройством; $0 \Rightarrow$ обмена нет
  - $\overline{ss}$ — отрицание сигнала ss
  - иногда вместо ss используется название cs: chip select, сигнал выбора схемы



SPI: miso

Попробуем соединить все порты *miso* настолько же “в лоб”, как и порты *mosi*

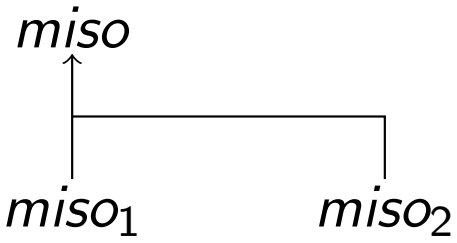


Значениями сигналов в портах *sclk*, $\overline{ss}$ и *mosi* управляет **одно** устройство M, а остальные “пассивно” считывают эти значения — так обычно и бывает в схемах

Возможность управлять значением на входе *miso* ведущего по необходимости (*если* $ss = 1$) должна быть предоставлена **каждому** ведомому устройству

Если попытаться реализовать передатчики ведомых устройств, используя **только** знания, полученные из всех предыдущих лекций, то возникнет непреодолимая трудность

SPI: miso



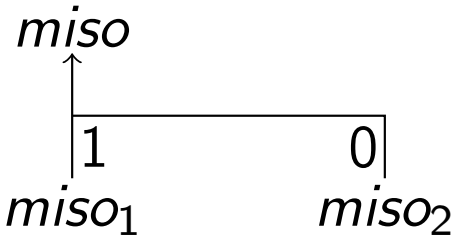
Непреодолимая трудность

miso в ведомом устройстве — *выход* схемы

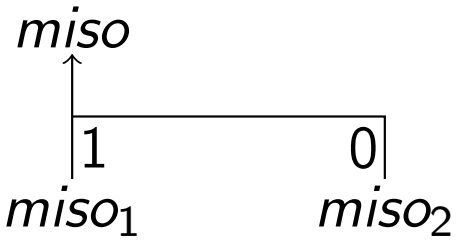
Согласно предыдущим лекциям, в каждый момент времени значение на этом выходе — либо 0, либо 1

Предположим, что

- если $ss = 0$, то в соответствующем ведомом устройстве $miso = 0$
- M обменивается данными с S_1
- S_1 передаёт значение 1



SPI: miso

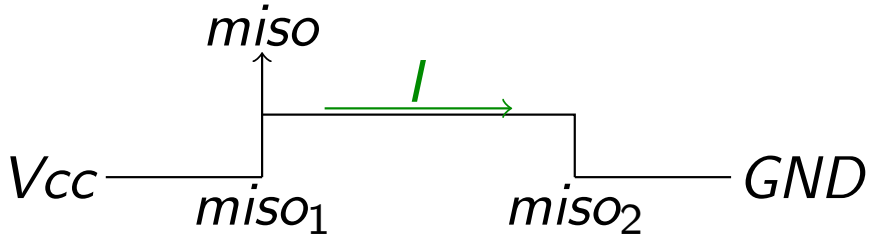


Непреодолимая трудность

Вспоминаем физику:

- “в точке x выставляется значение 0” = “точка x соединена проводником с источником постоянного напряжения GND”
- “в точке x выставляется значение 1” = “точка x соединена проводником с источником постоянного напряжения Vcc”

Результат — **короткое замыкание** в ведомых устройствах:



Чтобы преодолеть эту трудность, достаточно научиться **изолировать** значение в заданной точке схемы (*на выходах* $miso_k$) от источников напряжений GND и Vcc

Состояние высокого импеданса (z-состояние)

z — состояние высокого импеданса, оно же высокоимпедансное, оно же высокоомное, оно же плавающее (floating), оно же просто z-состояние

Это третье "логическое" значение, используемое в схемах наряду с 0 и 1

Точный смысл этого значения в точке x: ***сопротивление*** R **между** x **и остальными точкам схемы** ***очень большое***

Насколько большое: ***ток*** I **между** x **и остальными точками схемы** ***пренебрежимо мал***

С поправкой на небольшую погрешность, это означает, что $R = \infty$ и $I = 0$ — то есть точка x **изолирована** от всех остальных точек схемы, и в частности, от источников напряжений GND и Vcc

Попробуем придумать какой-нибудь "логический" элемент, способный выдавать на выходе значение z

Состояние высокого импеданса (z-состояние)

Спустимся сверху вниз по уровням абстракции описания схемы:

1. Схема — это описание способа реакции одних сигналов на изменение других сигналов (*Verilog*)
2. Схема — это набор регистров и система булевых функций, описывающих взаимосвязь значений на входах и выходах схемы и регистров (*RTL*)
3. Схема — это набор триггеров и логических вентилей (*последовательные схемы* и *комбинационные схемы с обратной связью*)
4. **Схема — это набор КМОП-транзисторов** (*КМОП-схемы*)

Из всех перечисленных уровней абстракции напряжения GND и Vcc упоминаются явно только на уровне **КМОП-схем**

Попробуем научиться изолировать точки схемы от GND и Vcc на этом уровне, и последовательно встроим то, чему научились, в остальные уровни

Z-состояние: КМОП и вентили

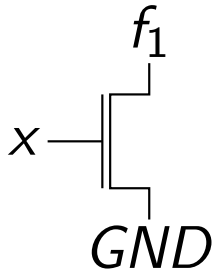
Рассмотрим несколько простых КМОП-схем, реализующих “логические” функции с z-состояниями в таблицах значений

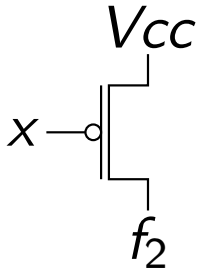
Начнём с функций с самой простой КМОП-реализацией:

x	f_1
0	z
1	0

x	f_2
0	1
1	z

Соответствующие КМОП-реализации:

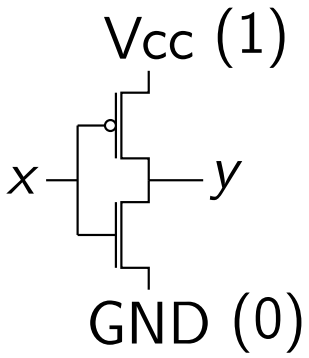




Z-состояние: КМОП и вентили

Напоминание: неконстантная булева функция с самой простой КМОП-реализацией — это отрицание:

x	$y = \overline{x}$
0	1
1	0

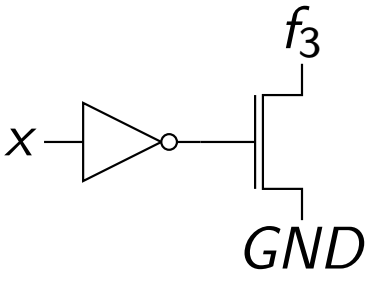


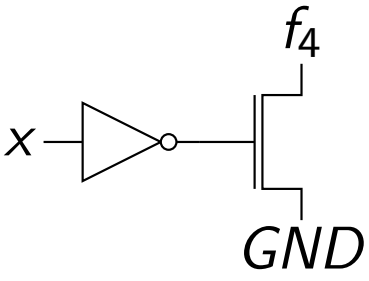
Попробуем реализовать такие функции:

x	f_3
0	0
1	z

x	f_4
0	z
1	1

Для этого достаточно совместить реализации функций f_1, f_2 и $\overline{x}$:

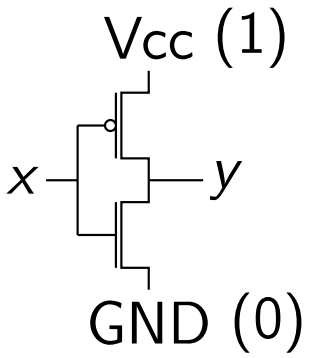




Z-состояние: КМОП и вентили

Напоминание: неконстантная булева функция с самой простой КМОП-реализацией — это отрицание:

x	$y = \overline{x}$
0	1
1	0

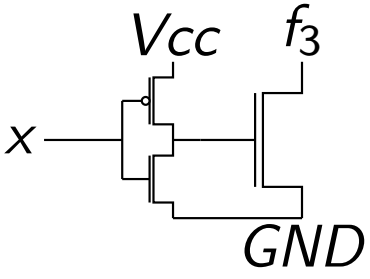


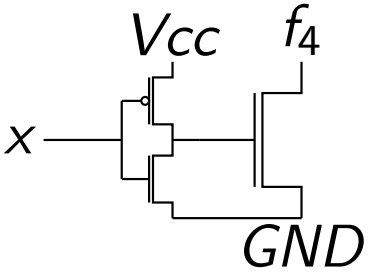
Попробуем реализовать такие функции:

x	f_3
0	0
1	z

x	f_4
0	z
1	1

Для этого достаточно совместить реализации функций f_1, f_2 и $\overline{x}$:



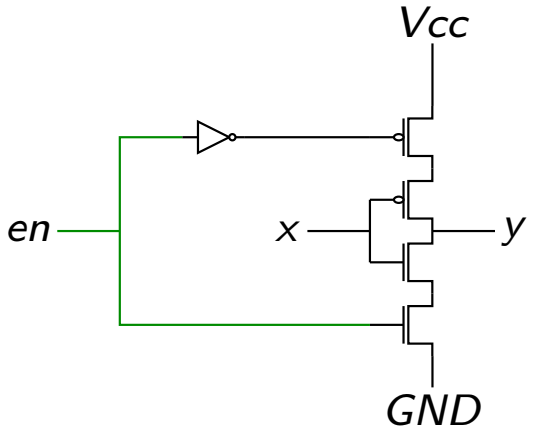


Z-состояние: КМОП и вентили

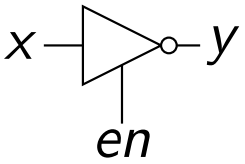
Рассмотрим функцию посложнее: отрицание с дополнительным входом, управляющим *изоляцией* выхода

$y = f_5(x, en)$		en	
		0	1
x	0	z	1
	1	z	0

КМОП-реализация:



Это управляемый инвертор, или инвертор с третьим состоянием (tri-state invertor)

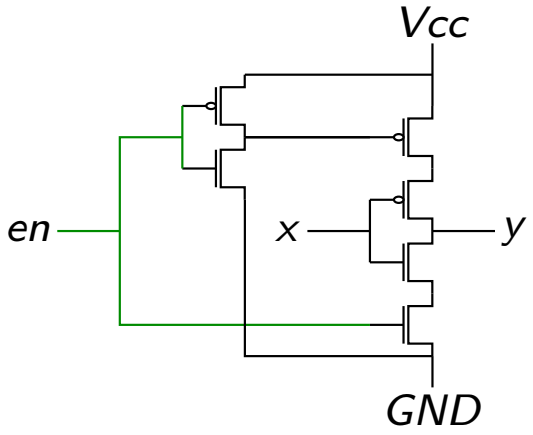


Z-состояние: КМОП и вентили

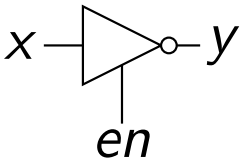
Рассмотрим функцию посложнее: отрицание с дополнительным входом, управляющим *изоляцией* выхода

$y = f_5(x, en)$		en	
		0	1
x	0	z	1
	1	z	0

КМОП-реализация:



Это управляемый инвертор, или инвертор с третьим состоянием (tri-state invertor)

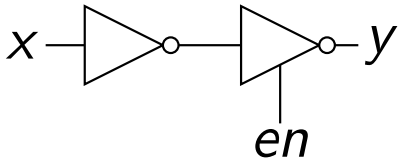


Z-состояние: КМОП и вентили

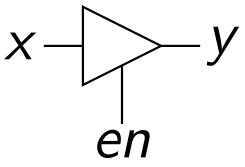
Попробуем аналогично расширить не отрицание, а тождественную функцию:

$y = f_6(x, en)$		en	
		0	1
x	0	z	0
	1	z	1

Эту функцию можно реализовать на уровне вентилей, не “спускаясь” на уровень КМОП-транзисторов:



Это управляемый буфер, или буфер с третьим состоянием (tri-state buffer)

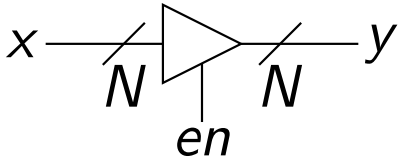


Z-состояние: RTL

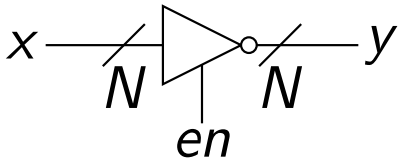
Вспоминаем, что такое RTL:

- Последовательная схема = **триггеры** + вентили
- RTL-схема = **регистры** + булевы функции
- Регистр = набор триггеров + входные шины + выходные шины

Управляемый буфер ширины N в зависимости от значения en либо изолирует выход y ширины N, либо направляет на него вход x ширины N



Управляемый инвертор ширины N в зависимости от значения en либо изолирует выход y ширины N, либо направляет в каждый разряд этого выхода отрицание соответствующего разряда на входе x ширины N



Z-состояние: Verilog

Программная семантика

`z` — такое же логическое значение, как и `0`, `1`, `x`

Правило расширения шины: если старший разряд расширяемой шины имеет значение `z`, то все добавляемые разряды имеют значение `z`

То есть "u = 1'bz" означает "все разряды точки u изолированы"

`casez` — команда, аналогичная *`casex`* и использующая `z` вместо `x` в качестве "любого" значения

Z-состояние: Verilog

Аппаратная семантика

Разрешено:

- *Как и для x:* **явно** использовать значение z в правых частях присваиваний
  - Если значение z используется неявно или комбинируется с другими значениями, то оно заменяется на x со всеми соответствующими разрешениями и запретами
- Использовать z в команде casez *так же, как x разрешено использовать в casex*
- Присваивать заданной точке значение **из нескольких процедур**, если *на уровне компиляции* достоверно известно, что в каждый момент времени значение, отличное от z, задаётся не более чем одной процедурой
  - Если какой-либо процедурой задаётся значение, отличное от z, то значения z, задаваемые остальными процедурами, игнорируются
  - Иначе итоговое значение в точке — z

Запрещено всё остальное

Z-состояние: Verilog

Примеры

```
wire [3:0] i, o; wire en;
assign o = en ? i : 1'bz;
```

```
wire [3:0] i; wire en;
reg [3:0] o;
always @*
if(en) o = i;
else o = 1'bz;
```

Это поддерживаемые реализации управляемого буфера ширины 4

Z-состояние: Verilog

Примеры

```
wire [1:0] en;
wire a, b, c;
assign c = (en == 2'd1) ? a : 1'bz;
assign c = (en == 2'd2) ? b : 1'bz;
```

Это поддерживаемый код, задающий значение в точке c согласно следующей таблице:

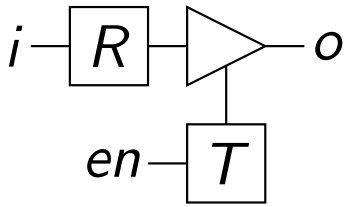
c		en			
		00	01	10	11
$a\ b$	00	z	0	0	z
	01	z	0	1	z
	10	z	1	0	z
	11	z	1	1	z

Z-состояние: Verilog

Примеры

```
wire [3:0] i; wire en;
reg [3:0] o;
always @(posedge clk)
if(en) o <= i;
else o <= 1'bz;
```

Это поддерживаемый код, который, *согласно написанному в стандарте*, может иметь следующую аппаратную семантику:



- R — параллельный регистр, запоминающий значение i (*если* $en = 0$, *то поведение* R *произвольно*)
- T — D-триггер, запоминающий значение en
- если в T сохранено значение 1, то o совпадает с выходом R, а иначе выход o изолирован

Z-состояние: Verilog

Примеры

```
reg a, b;
always @* begin
a = 1'bz;
b = a;
end
```

Это **неподдерживаемый** код: значение z в присваивании для b используется **неявно**

Z-состояние: Verilog

Примеры

```
wire a, b, c;
assign a = 1'bz;
assign c = a && b;
```

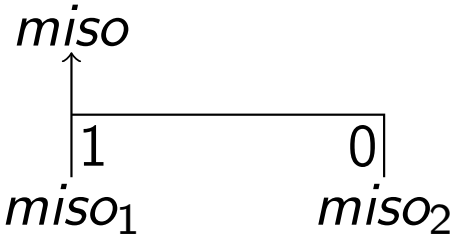
Это **неподдерживаемый** код: значение z в присваивании для c

- используется **неявно** и
- **комбинируется** со значением b

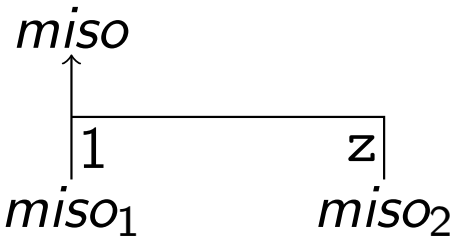
SPI: miso

Теперь можно легко преодолеть *трудность*, связанную с выставлением сигнала в одном проводе из нескольких точек:

- Если M обменивается данными с устройством S, то S *подключено* к проводу и выставляет значения 0, 1 согласно протоколу
- Если M не обменивается данными с S, то S *изолирует* выход *miso* от своей основной подсхемы



— короткое замыкание



— типичный способ подключения устройств к общему соединению для поочерёдного выставления логических значений

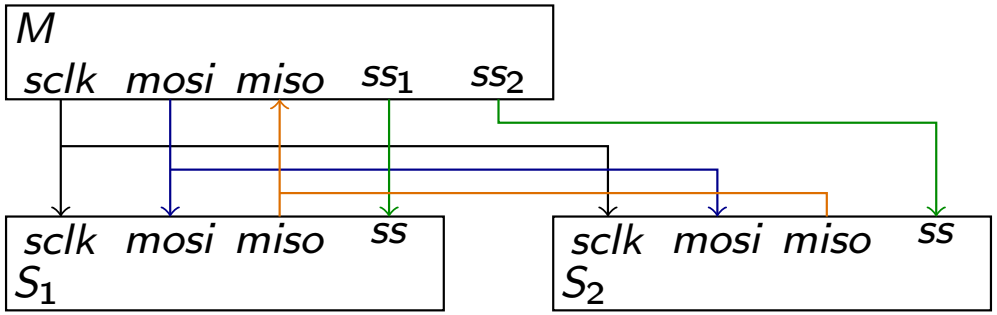
Такое общее соединение обычно называется шиной[1] и магистралью

[1] Я уже упоминал, что у термина “шина” много значений

SPI: соединение портов

Итоги

Все порты в протоколе SPI можно соединить “в лоб”:



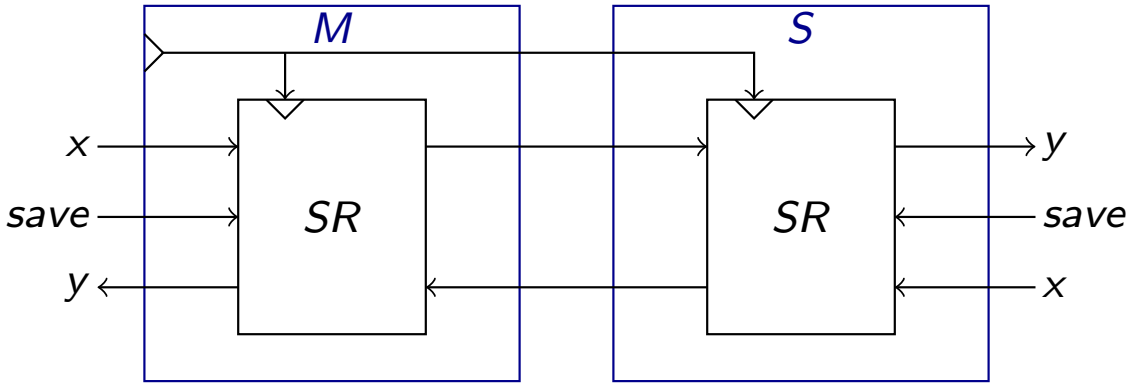
(*давайте для простоты забудем про то, что для выбора ведомого устройства используется порт $\overline{ss}$, а не ss*)

Если $ss = 1$, то выход *miso* соединён со схемой соответствующего ведомого устройства S_i, и S_i передаёт разряды сообщения согласно протоколу SPI для двух устройств

Если $ss = 0$, то выход *miso* изолирован от источников напряжения ведомого устройства S_i, и S_i не передаёт разряды сообщения

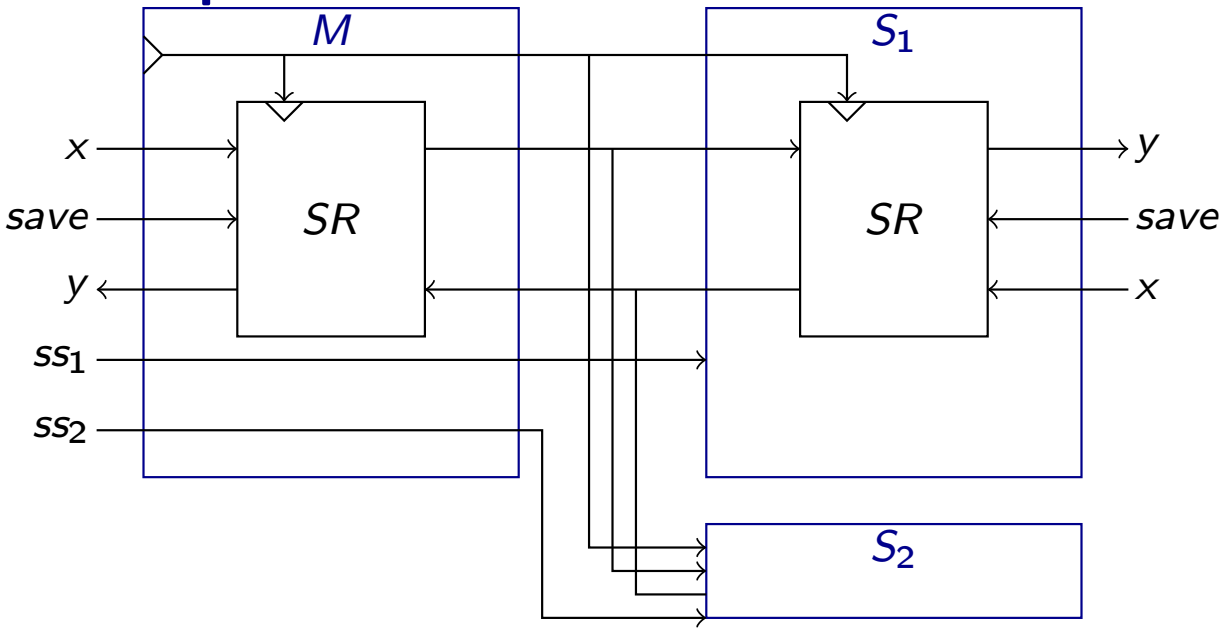
SPI: реализация

Вспомним, как выглядела реализация приёмника и передатчика по протоколу SPI для двух устройств:



Адаптируем эту реализацию к общему случаю (*вернее, к рассматриваемому упрощённому случаю с двумя ведомыми устройствами*)

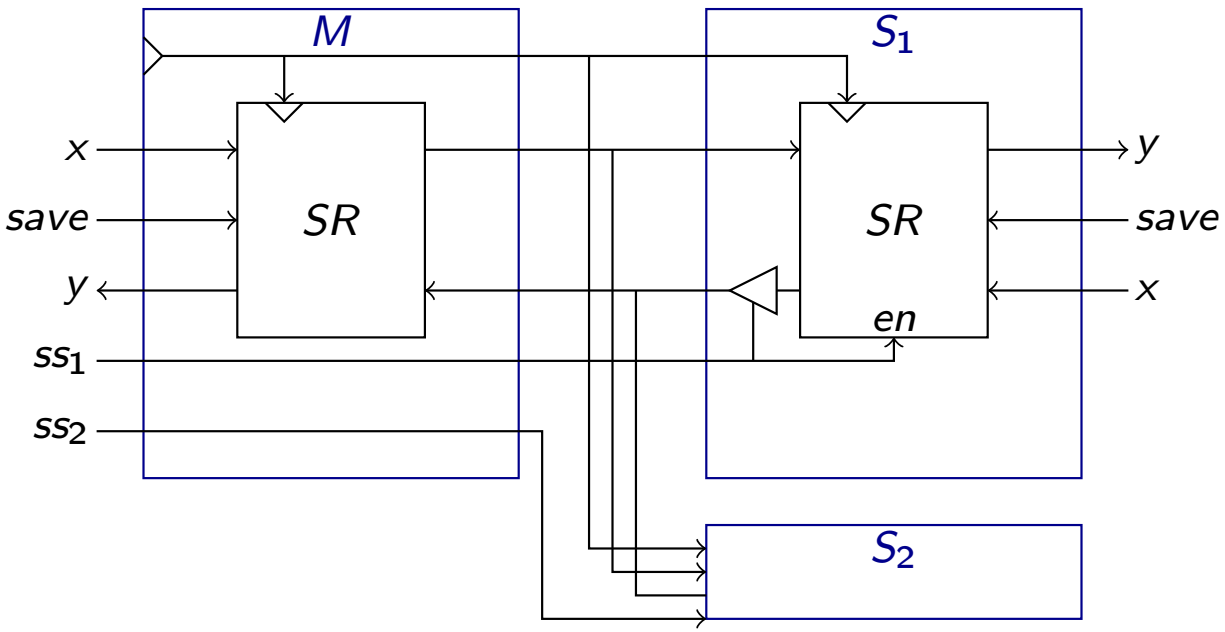
SPI: реализация



Будем полагать, что сигналы ss_1 и ss_2 ведущего устройства поступают на входы схемы M

Тогда можно легко модифицировать схему для случая нескольких ведомых устройств:

- добавим в сериализатор SR вход en:
  - $en = 1 \Rightarrow$ сериализатор работает как обычно
  - $en = 0 \Rightarrow$ сериализатор не свдигает хранящееся число
- добавим в ведомое устройство *управляемый буфер*, подходящим образом изолирующий и подключающий выход *miso*

SPI: реализация

Итог: