Программируемые логические интегральные схемы

(спецкурс)

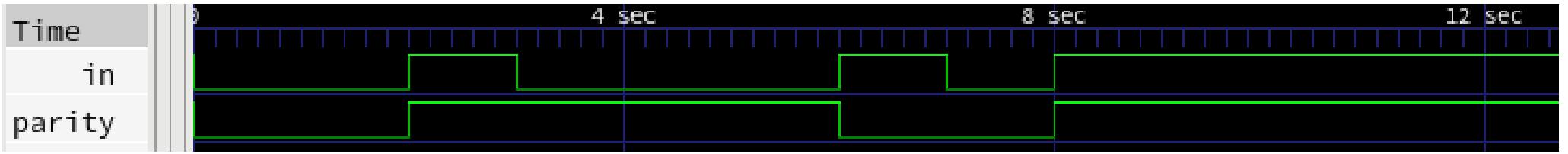
2017, весенний семестр

Чем будем заниматься?

Учиться разрабатывать **схемы**, управляющие аппаратурой

Управление происходит выставлением значений
- 0 (низкий уровень напряжения) и
- 1 (высокий уровень напряжения)

в реальном времени в нужных проводах



Пример: счётчик чётности для события “сигнал in изменяется с 0 на 1”

in – входной сигнал

например, поступающий с кнопки, включающей и выключающей лампочку

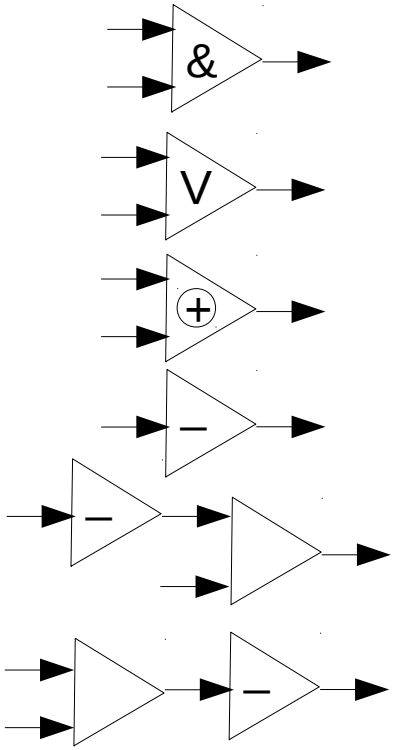
parity – выходной сигнал, вырабатываемый схемой

например, управляющий цепью подачи напряжения на лампочку

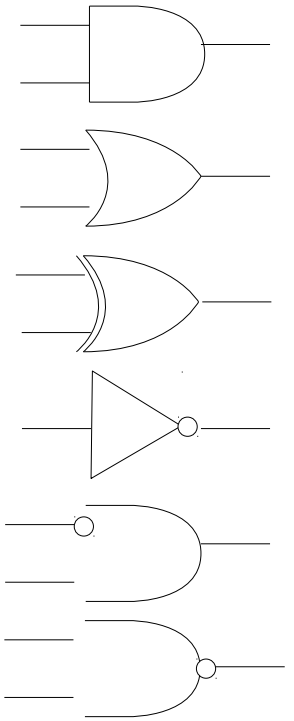
Основные примитивы схем

Комбинационные элементы

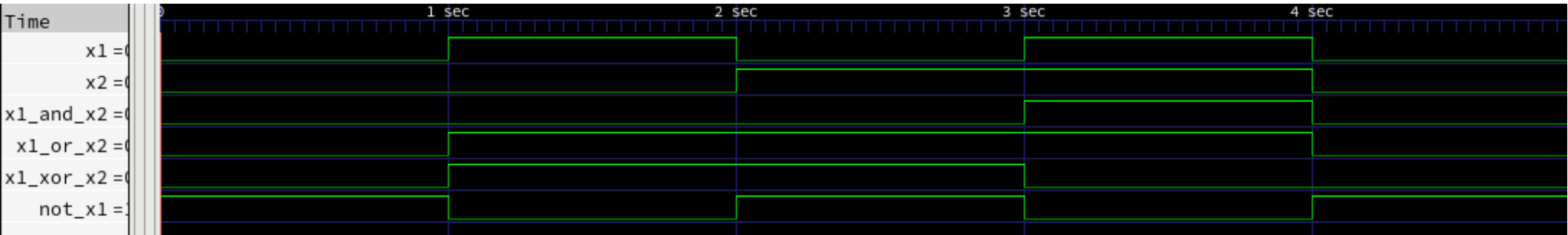
Как это было в СФЭ



Как это выглядит в комбинационных схемах

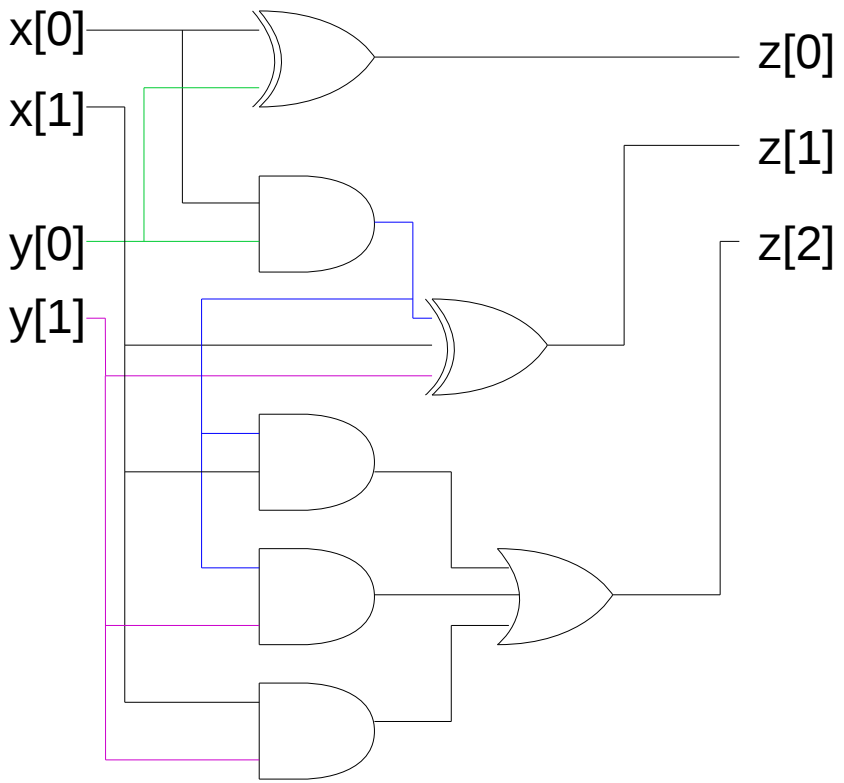


Как это работает



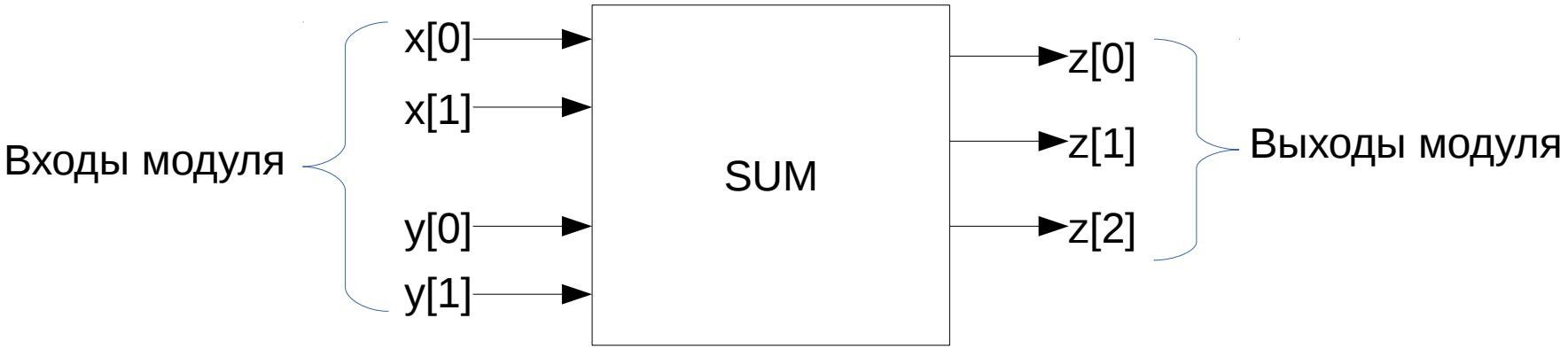
Основные примитивы схем

Комбинационные элементы можно соединять в комбинационные схемы так же, как и функциональные элементы в СФЭ



Основные примитивы схем

Логически целостные части проектируемой схемы
образуют модули (“чёрные ящики”)
на более высоком уровне абстракции при проектировании

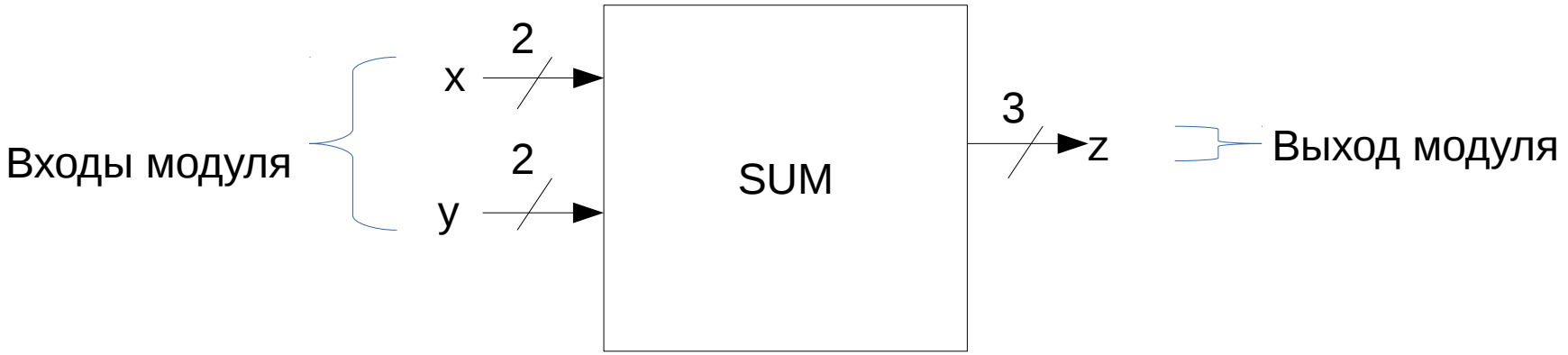


Часто на входах и выходах модулей располагаются структуры,
представляющие собой массивы (здесь принято говорить “**шины**”) данных

(*например, здесь – три шины, содержащие двоичные записи чисел x, y, z*)

Основные примитивы схем

Логически целостные части проектируемой схемы
образуют модули (“чёрные ящики”)
на более высоком уровне абстракции при проектировании



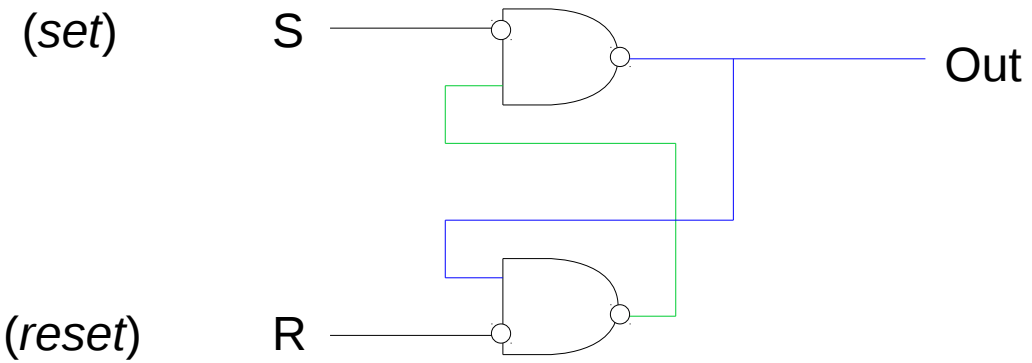
Часто на входах и выходах модулей располагаются структуры,
представляющие собой массивы (здесь принято говорить “**шины**”) данных

(*например, здесь – три шины, содержащие двоичные записи чисел x, y, z*)

Для обозначения шин данных используется особая графическая нотация

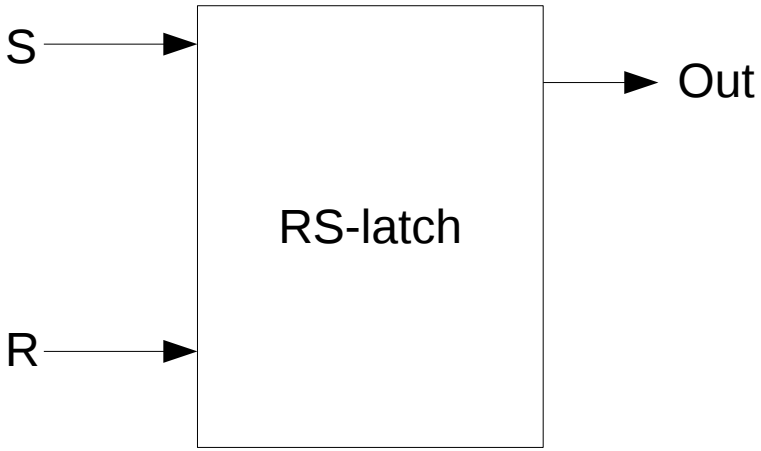
Основные примитивы схем

RS-защёлка (*или RS-триггер*)



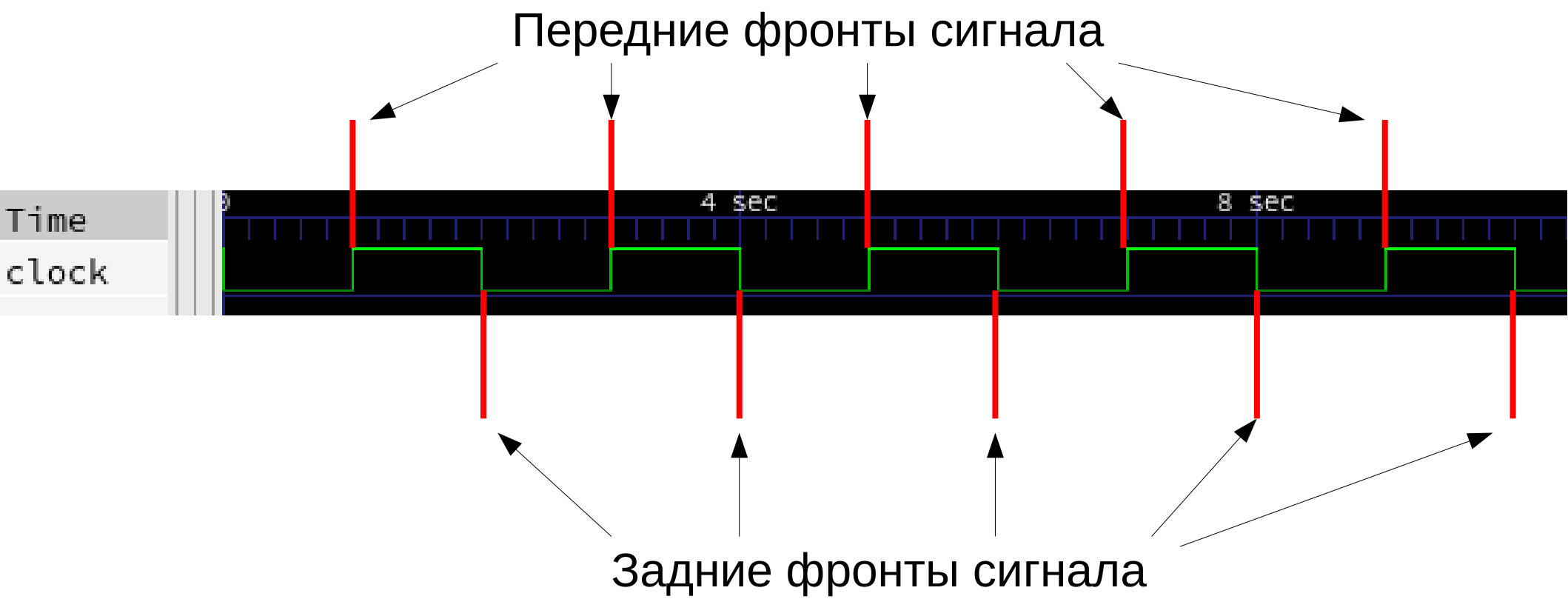
Основные примитивы схем

RS-защёлка (*или RS-триггер*)



Основные примитивы схем

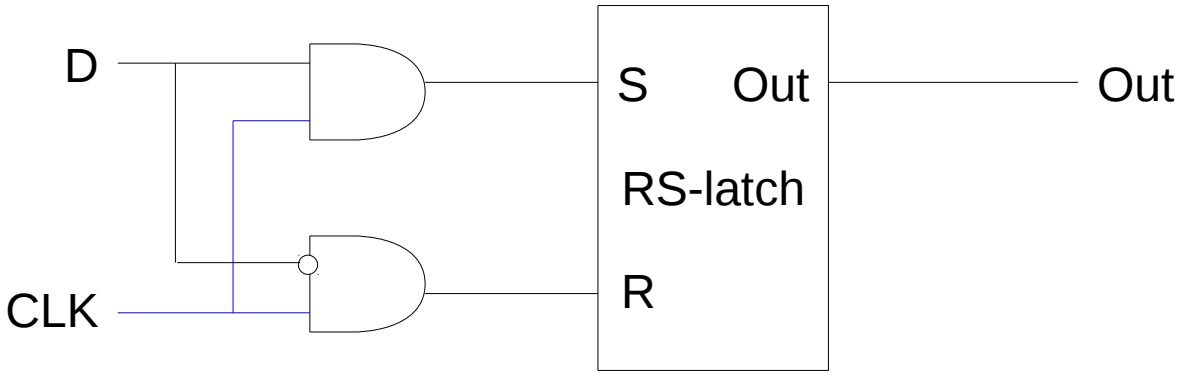
Тактовый сигнал

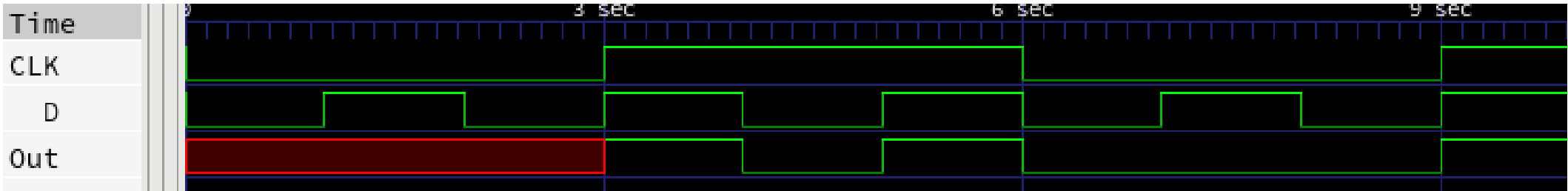


Основные примитивы схем

D-защёлка

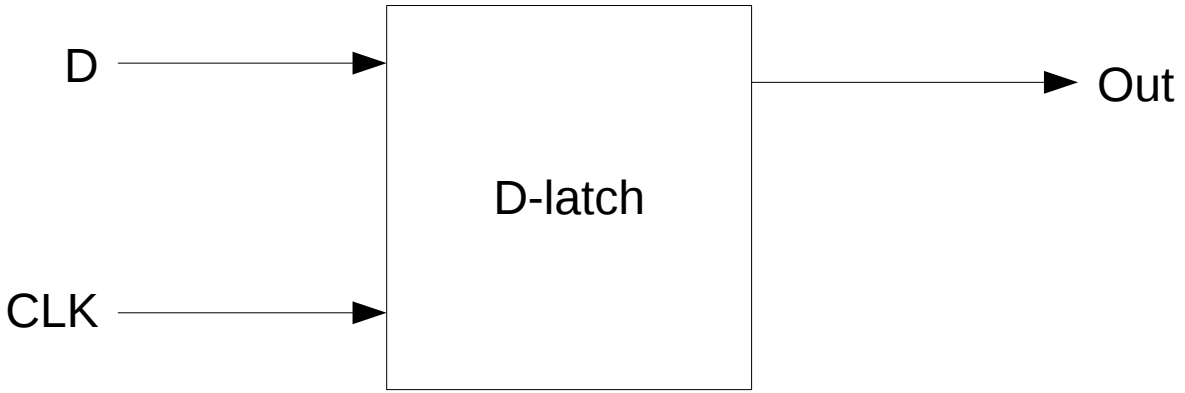
(*delayed input*)

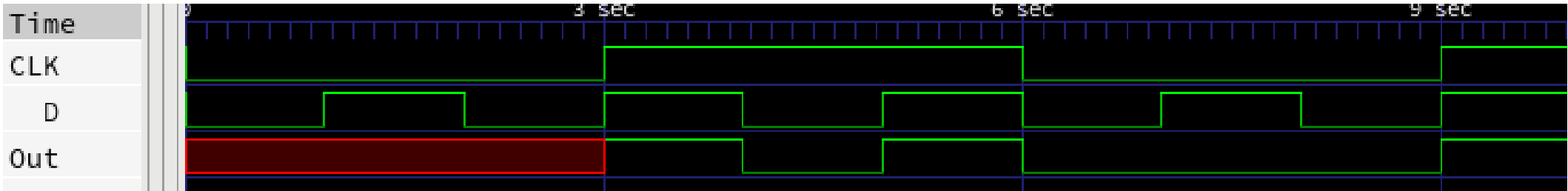




Основные примитивы схем

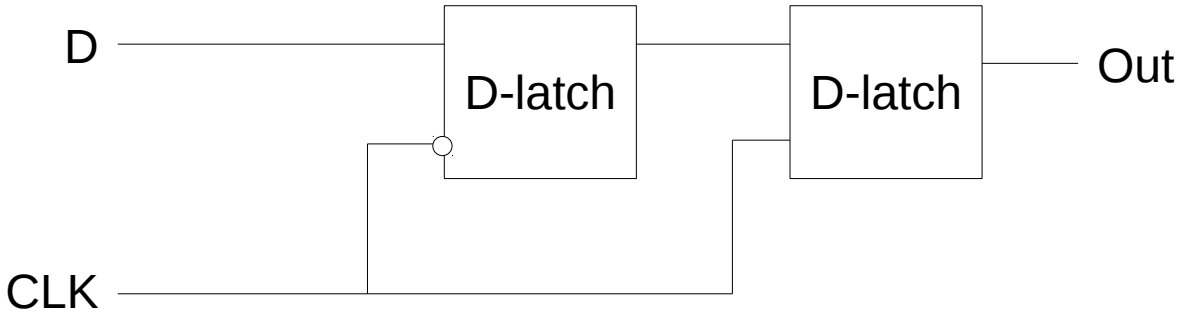
D-защёлка

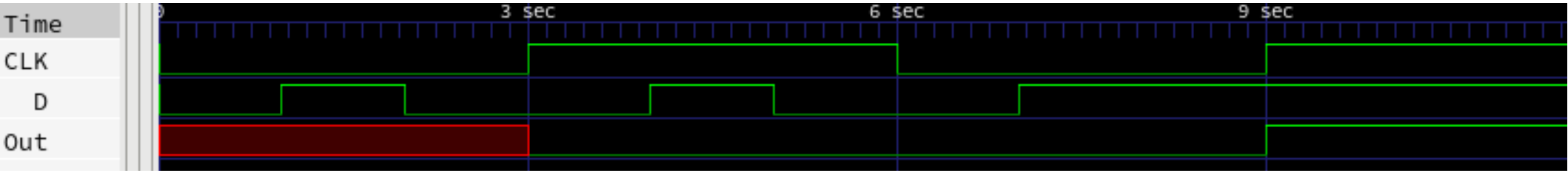




Основные примитивы схем

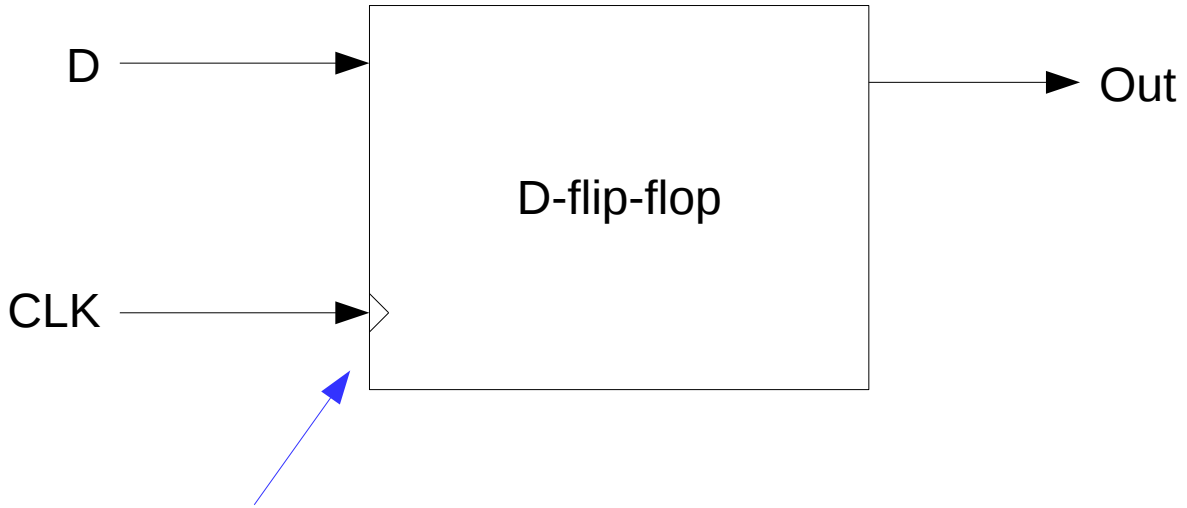
D-триггер



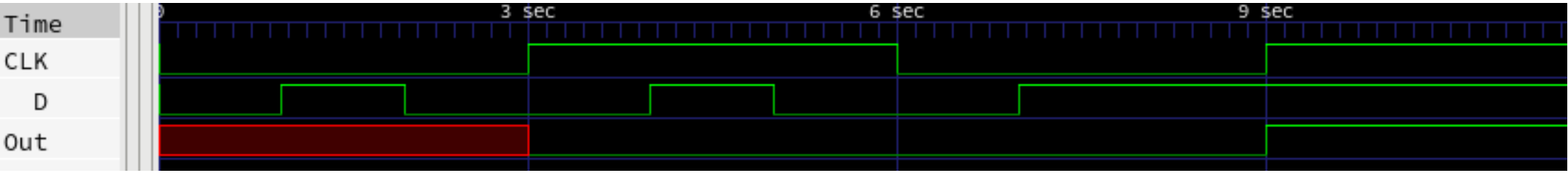


Основные примитивы схем

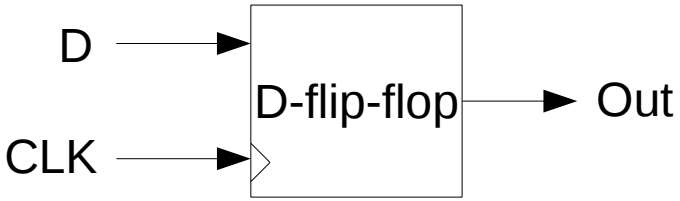
D-триггер



Этот треугольник означает, что модуль реагирует на фронт сигнала



Основные примитивы схем



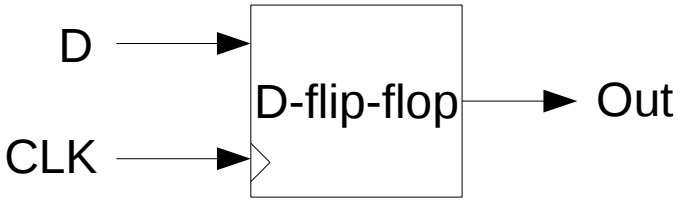
D-триггер – это наиболее часто встречающаяся реализация **ячейки памяти** в схеме

Если попытаться коротко описать работу D-триггера с учётом знаний, имеющихся из курса “Дискретная математика”, то описание выйдет такое:

D-триггер – это “реальная” схема, дискретной моделью которой является **элемент единичной задержки** в СФЭЗ

Единственное функциональное отличие D-триггера от элемента задержки состоит в том, что *абстрактное дискретное* время заменяется на дискретный подсчёт передних фронтов *реального* тактового сигнала

Основные примитивы схем



“Реальный” аналог СФЭ - это **комбинационная схема**

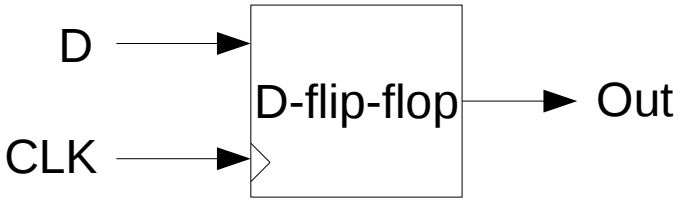
(*более точно, это “реальная” схема без памяти:*
в каждый момент времени выходы зависят
только от входов в тот же момент времени)

“Реальный” аналог СФЭЗ - это **последовательная схема**

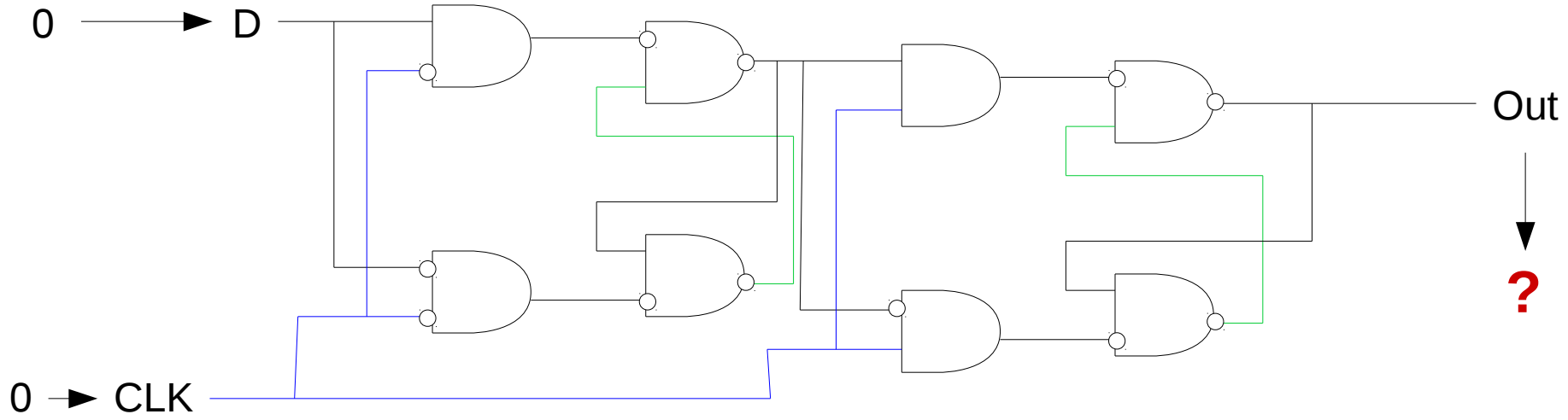
(*более точно, это “реальная” схема памятью*)

А что такое “тот же момент времени”?

Основные примитивы схем



А какое значение выдаётся на выход D-триггера до самого первого переднего фронта тактового сигнала?



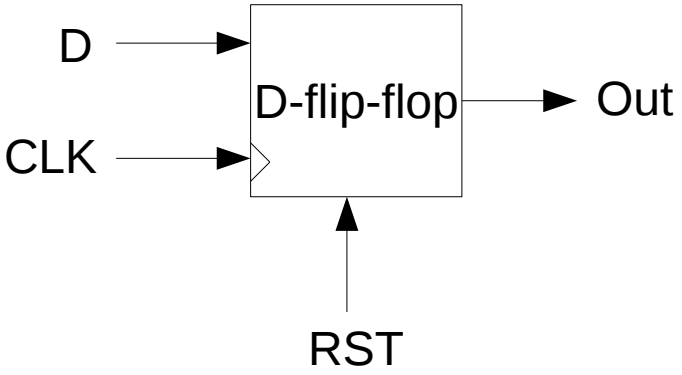
Чтобы логические элементы заработали как надо, на них должно быть подано напряжение

В зависимости от того, в каком порядке подавалось напряжение
и как элементы реагировали на “плохие” напряжения,
в конечном итоге на выходе до первого фронта CLK может оказаться **любое** значение

Более того, в реальных схемах память *вряд ли* будет реализована именно так

Основные примитивы схем

D-триггер со сбросом



Чтобы исключить неопределённость начального значения выхода триггера, проектировщики схем часто вводят два сигнала синхронизации в схеме:

- тактовый сигнал (CLK)
- сигнал сброса (RST)

По переднему фронту сигнала сброса D-триггер *со сбросом* запоминает и выдаёт на выход значение “0”

Сигнал сброса, как и тактовый сигнал, может быть подан на все элементы памяти схемы, и в момент сброса схема может считаться полностью инициализированной

Ещё о тактовом сигнале

А что такого особенного в тактовом сигнале?

На самом деле тактовый сигнал – это точно такой же сигнал, как и все остальные

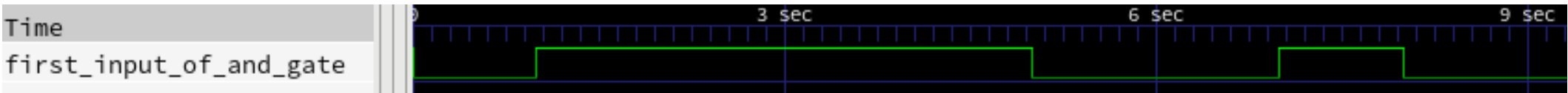
Иначе говоря, **любой** сигнал может выступать в роли тактового

Обычно сигнал называют тактовым, если с его помощью синхронизируется работа нескольких концептуально независимых частей схемы
(как правило, с помощью подсчёта передних фронтов)

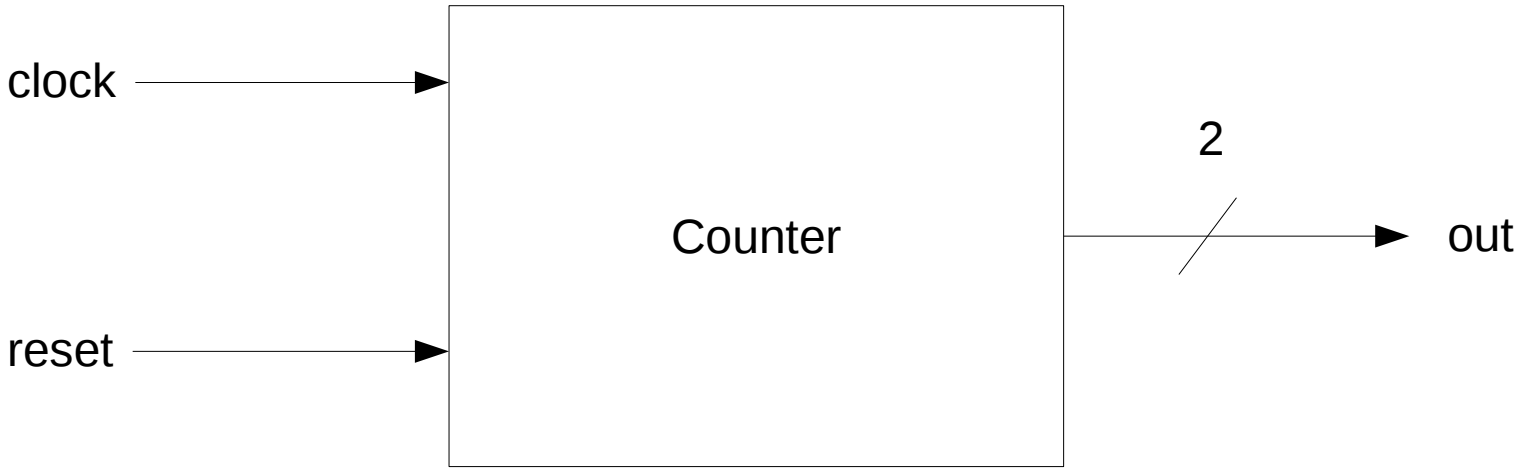
Правило хорошего тона:

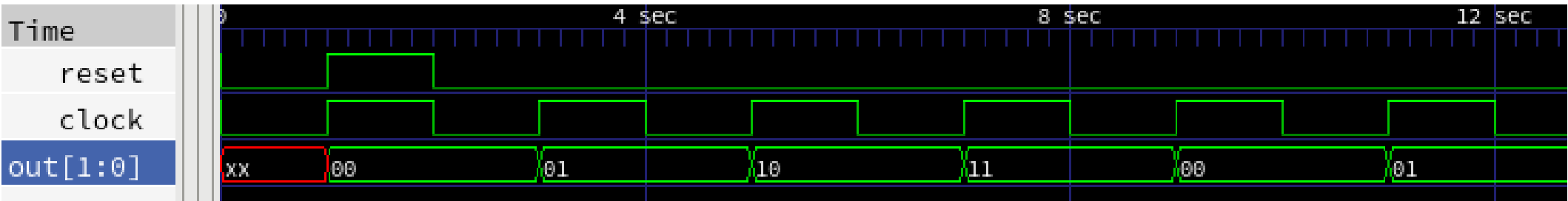
стараться выдерживать заданную частоту осцилляции тактового сигнала

Но по большому счёту, даже такой сигнал можно считать тактовым,
если мы с его помощью синхронизируем разные части схемы:

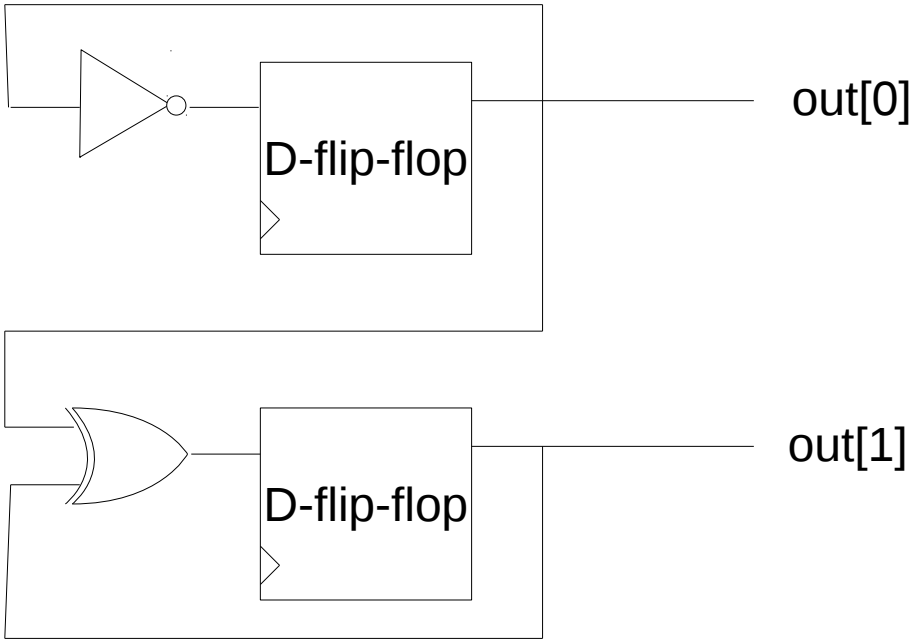


Пример: двухбитовый счётчик

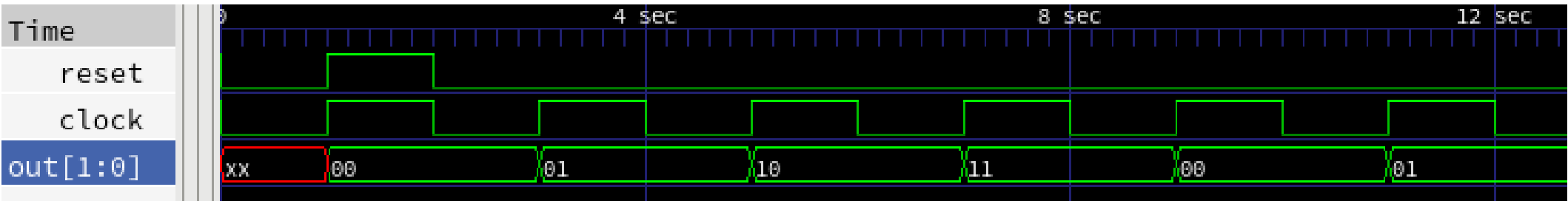




Пример: двухбитовый счётчик



Сигнал CLK и сигнал RST, направленные на входы обоих триггеров, опущены для повышения читаемости схемы



Пример: двухбитовый счётчик

Как это выглядит на плате

Требования по технике и программам

По желанию можем попробовать организовать работу с платами в компьютерном классе

Если захотите учиться писать схемы самостоятельно, то вам понадобятся:

- Quartus Prime software Lite edition (включая ModelSim)

  для работы непосредственно с платой DE0-Nano

  можно скачать с сайта *altera.com*

- Icarus Verilog

  для более быстрой работы со схемами, когда под рукой нет платы

  любая система: скачать с сайта *iverilog.icarus.com*

  Linux: поставить пакет iverilog из официального репозитория

- GtkWave

  для удобной визуализации сигналов при работе с Icarus Verilog

  любая система: скачать с сайта *gtkwave.sourceforge.net*

  Linux: поставить пакет gtkwave из официального репозитория

Некоторые полезные ресурсы

mk.cs.msu.ru → Спецкурсы → Программируемые логические интегральные схемы

Это страница курса, на ней будет много всего полезного,
и в частности, выложены слайды, по которым рассказывалась
первая часть спецкурса в осеннем семестре
(в текущей второй части некоторые моменты будут повторяться)

mk.cs.msu.ru → Семинары →Практикум по пакетам проектирования сверхбольших интегральных схем (осенний семестр 2016 года)

Это страница практикума для магистров второго года обучения,
на основе которого создавался эта упрощённая версия: спецкурс

asic-world.com

Это ресурс, на котором некоторые проектировщики схем
выкладывают пояснения и инструкции по всему, по чему только можно
(в частности, доходчиво объясняют всё про Verilog)

edaplayground.com

Некоторые студенты,
не желающие настраивать у себя на компьютере нужные программы,
были замечены разрабатывающими схемы в web-интерфейсе этого сайта

Как это всё выглядит в Verilog

Синтаксис языка Verilog описан в слайдах **на страницах спецкурса и практикума**

Домашнее задание:

- поставить Quartus, Icarus Verilog и GtkWave
- посмотреть в слайды с синтаксисом языка Verilog на страницах спецкурса и практикума и разобраться в том, как устроен Verilog
- чтобы проверить своё понимание синтаксиса, можно попытаться написать модуль с однобитовыми входами x1, x2 и выходом o, считающий чётность числа событий “сигнал x1 переключился и совпал с сигналом x2” и выслать решение на почту
- (*по желанию*) можно попробовать получить осцилограммы, однако про симуляцию и их получение скорее всего будет одно из дальшейших занятий