Языки описания схем

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

Блок K2

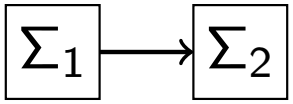
Кое-что ещё:
Протокол UART (схемная реализация)

Лектор:
Подымов Владислав Васильевич

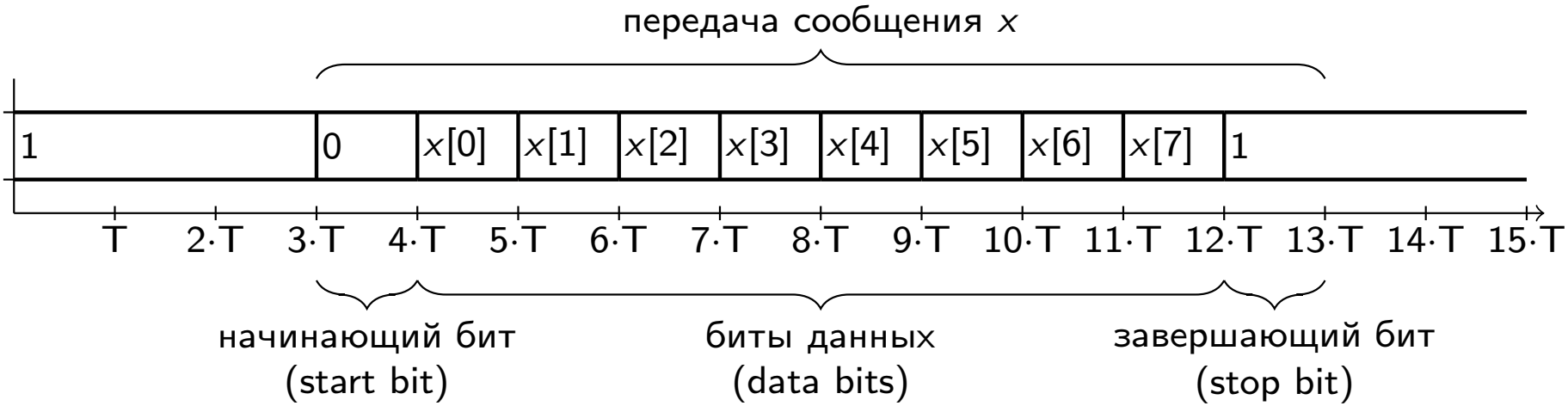
E-mail:
valdus@yandex.ru

ВМК МГУ, 2020, осенний семестр

Вступление



Передача одного сообщения x по протоколу *UART* (самый простой вариант):



$\nu = \frac{1}{T}$ — *частота протокола*

В схемной реализации передачи сообщений по протоколу UART содержатся две независимые части:

- реализация передачика (Σ_1)
- реализация приёмника (Σ_2)

Попробуем реализовать Σ_1 и Σ_2 как *синхронные схемы со сбросом*

UART: как реализовать передатчик

Рассмотрим *простой* вариант передатчика:

- Порты:
  - *clk*, *rst* — тактовый сигнал и асинхронный сброс
  - вход *x* ширины 8
  - вход *start* ширины 1
  - выход *tx* ширины 1
    - Transmit Data $\xrightarrow{x?}$ TxD $\rightarrow$ Tx
      Это устоявшееся обозначение прóвода передачи данных по UART
- После сброса в *tx* выдаётся тишина (1), пока не будет прочитано значение $start = 1$
- После чтения значения $start = 1$ передатчик
  - сохраняет и поразрядно выдаёт в *tx* сообщение *x*, игнорируя значения *start*, и после этого
  - сбрасывается

Попробуем придумать схему такого передатчика —
может быть, не самую лучшую, но какую-нибудь попроще

UART: как реализовать передатчик

Операционный автомат

Основное, что должен уметь делать операционный автомат передатчика:

- сохранять значение со входа x, и
- **последовательно** выдавать в tx биты сообщения: начинающий бит, затем биты сохранённого значения, и затем завершающий бит

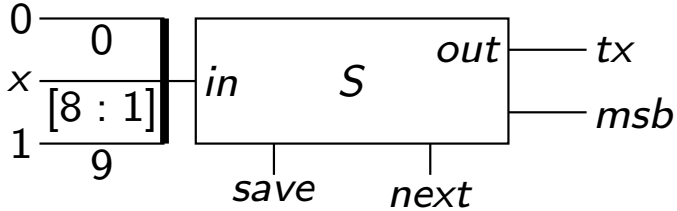
Схема, умеющая сохранять значение и последовательно выдавать его разряды, уже встречалась в практической части курса под названием "*сериализатор*"

UART: как реализовать передатчик

Операционный автомат

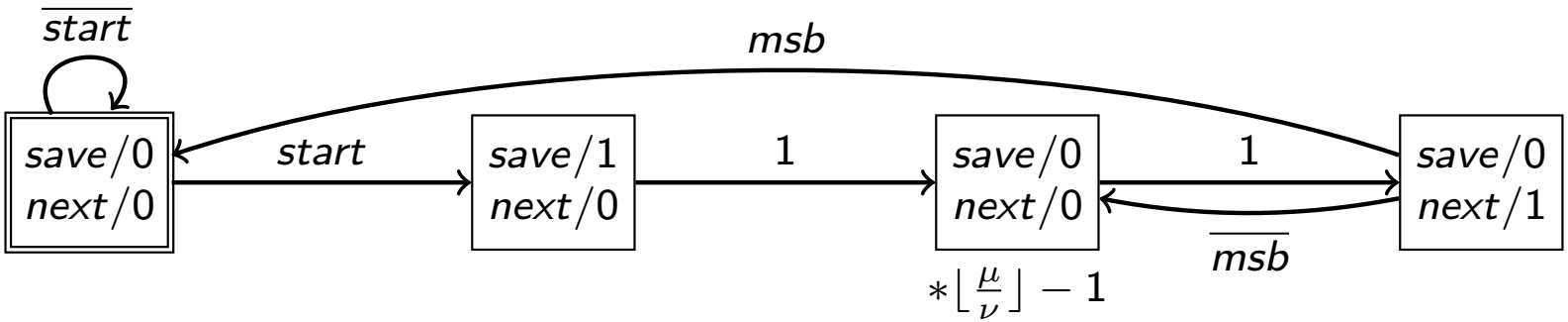
Используем в операционном автомате такой сериализатор S:

- Порты:
  - вход *in* ширины 10
    (*биты сообщения, включая начинающий (0-й) и завершающий (9-й)*)
  - входы *save* и *next* ширины 1
  - выходы *out*, *msb* ширины 1
- В сериализаторе хранятся значение v ширины 10 (сообщение) и значение i ширины 3 (номер передаваемого разряда)
- $out = v[i]$ (если $i > 9$, то $out = 1$)
- $msb = 1 \Leftrightarrow i = 9$
- Сброс $\Rightarrow$ сохраняется значение $i = 10$
- $save = 1 \Rightarrow$ сохраняются значения $v = in$ и $i = 0$
- $save = 0$, $next = 1 \Rightarrow v$ не изменяется, i увеличивается на 1
- $save = next = 0 \Rightarrow v$ и i не изменяются



UART: как реализовать передатчик

Управляющий автомат:

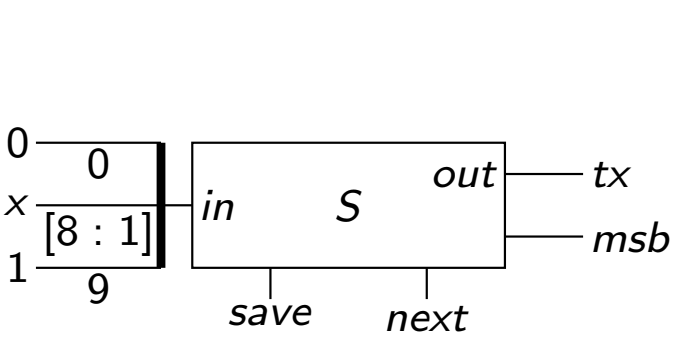


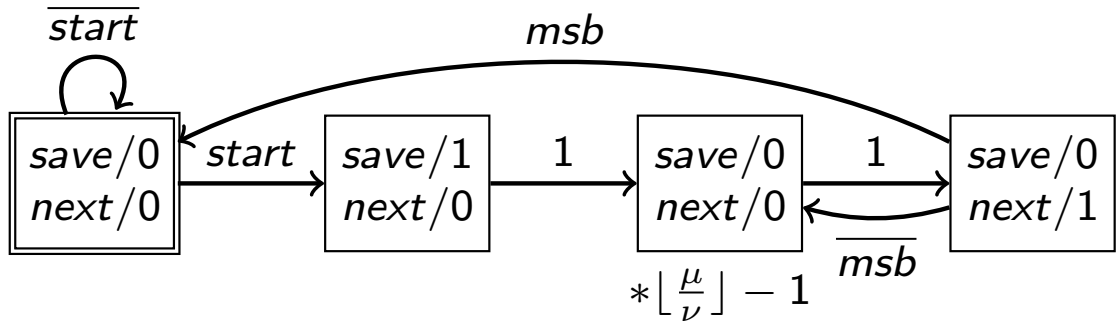
Состояния слева направо: (μ — частота передатчика)

1. Выдаём значение $tx = 1$, не изменяем состояние S
2. Готовимся к пересылке: при переходе к следующему такту сохраняется сообщение (1 x 0), и в tx выдаётся 0-й разряд
   - *Появился лишний такт между "start = 1" и началом передачи — для простоты оставим всё как есть*
3. Выдаём бит сообщения *приблизительно* $(T - \frac{1}{\mu})$ единиц времени
4. Последний такт выдачи бита сообщения (оставшийся период $\frac{1}{\mu}$):
   - Если выдавался последний бит значения сериализатора, то завершаем передачу
   - Иначе начинаем выдачу следующего бита

UART: как реализовать передатчик

Итог:





Эту схему можно легко адаптировать и к другим вариантам протокола:

- *Частота протокола:* учтена в числе $\frac{\mu}{\nu}$
- *Ширина пересылаемого значения:* выбрать подходящую ширину S
- *Порядок пересылки разрядов:*
  подходящим образом соединить шину x с портом *in*
- *Механизмы обеспечения корректности:*
  расширить S требуемыми разрядами
  и соединить их с требуемыми значениями
  (1, сумма разрядов x, отрицание суммы разрядов x)

UART: как реализовать приёмник

Схема передатчика оказалась простой: всё, что она должна уметь —

- начинать передачу согласно заданному сигналу
- сохранять значение на входе в начале передачи
- поразрядно выдавать сообщение
  с заданным числом тактов на разряд

Приём сообщения устроен более сложно:

- часто́ты и фазы тактовых сигналов передатчика и приёмника независимы и
  никак не могут быть синхронизированы внутри приёмника
- сообщение должно корректно приниматься с учётом всех *погрешностей* и независимо от сочетания частот и фаз

UART: как реализовать приёмник

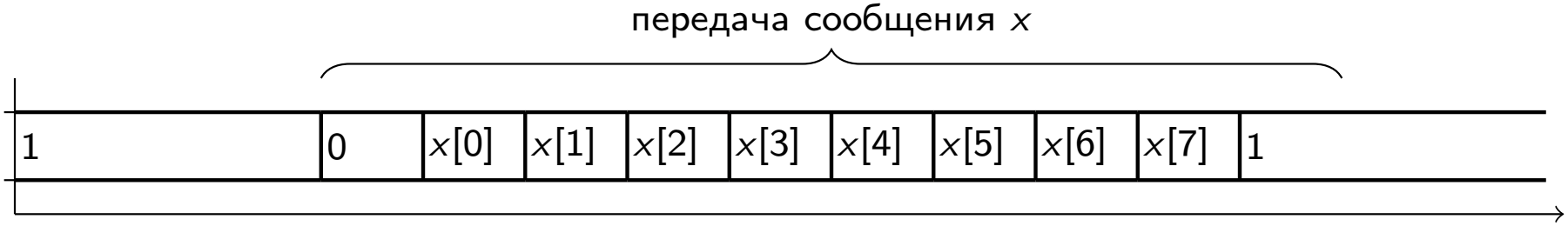
Рассмотрим *простой* вариант приёмника:

- Порты:
  - *clk*, *rst* — тактовый сигнал и асинхронный сброс
  - выход *x* ширины 8
  - выход *busy* ширины 1
  - вход *rx* ширины 1
    - Receive Data $\xrightarrow{x?}$ RxD $\rightarrow$ Rx

      Это устоявшееся обозначение прóвода приёма данных по UART
- $busy = 1 \Leftrightarrow$ сообщение принимается
- После приёма сообщения и до начала следующей передачи в *x* выводится принятое число

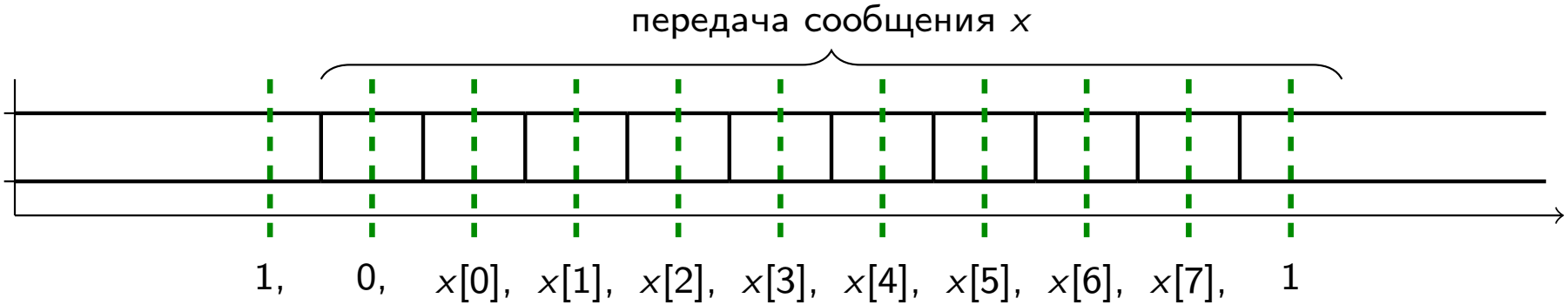
Попробуем придумать схему такого приёмника —
может быть, не самую лучшую, но какую-нибудь попроще

UART: как реализовать приёмник



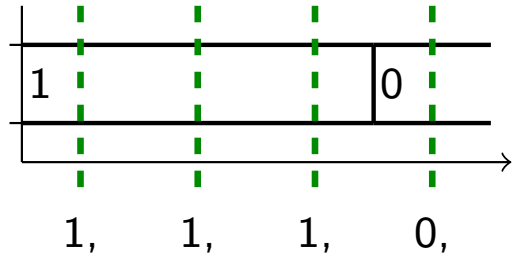
Приёмник, как и любая синхронная схема, может читать значения со входов и изменять состояние **только** в моменты выбранных фронтов тактового сигнала

Как следствие, в приёмнике должно выполняться подходящее семплирование (дискретизация) входа *rx*: замена непрерывного сигнала на дискретную последовательность цифровых значений этого сигнала в выбранные моменты времени



UART: как реализовать приёмник

Начало приёма сообщения



Пусть μ — частота приёмника, и $\tau = \frac{1}{\mu}$

Если приёмник считывал со входа тишину (1), и затем по очередному фронту тактового сигнала в момент t считал значение 0, то он может быть уверен в том, что

- передача сообщения началась, и
- момент начала передачи сообщения лежит в интервале $[t - \tau, t]$

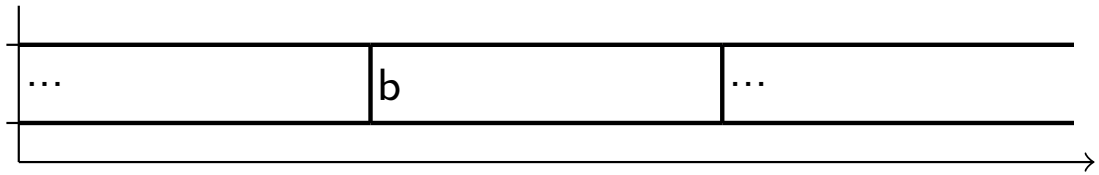
Назовём фронт, по которому считано такое значение 0, нолевым, и начнём с него отсчёт фронтов

Основная задача приёмника после нолевого фронта — "аккуратно" принять все биты данных

UART: как реализовать приёмник

Приём очередного бита

Рассмотрим *идеальную* пересылку одного бита сообщения (b):



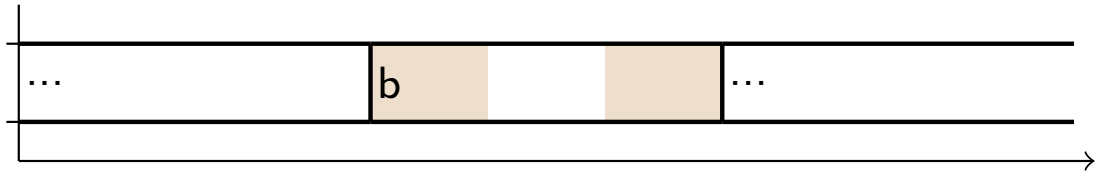
Предположим, что известно точное время t начала пересылки сообщения

Тогда можно точно вычислить *идеальное* время начала (ℓ) и конца (r) пересылки бита b, отталкиваясь от t и частоты протокола

В реальности длительность пересылки каждого бита колеблется в пределах некоторой *погрешности*, и погрешность может накапливаться с каждым следующим битом

UART: как реализовать приёмник

Приём очередного бита



Из-за погрешностей в интервале пересылки каждого бита появляются “серые зоны”: интервалы $[\ell, \ell + \Delta]$ и $[r - \Delta, r]$, где Δ — наибольшая возможная накопленная погрешность пересылки бита

Попытка считать значение текущего бита из серой зоны может привести к тому, что из-за накопленной погрешности фактически считается значение предыдущего или следующего бита

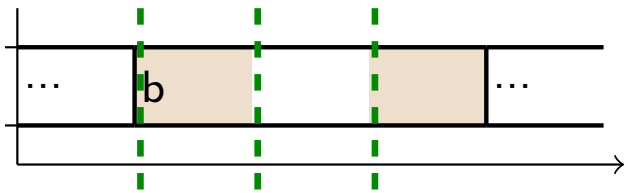
Семплирование сообщения приёмником должно быть устроено так, чтобы значение каждого бита читалось в *идеальном* интервале, но **вне серой зоны**

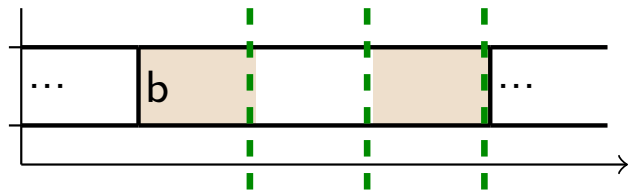
UART: как реализовать приёмник

Приём очередного бита

Положим для ясности, что частота приёмника в три раза выше частоты протокола

Тогда в каждом *идеальном* интервале пересылки каждого бита содержится **ровно три** фронта с такими граничными случаями их расположения (в зависимости от точного времени начала передачи сообщения):





Если ширина серой зоны с каждой стороны меньше трети интервала пересылки b, то

- средний фронт обязательно находится вне серой зоны
- средний фронт для бита $x[i]$ имеет номер $3 \cdot i + 4$

UART: как реализовать приёмник

Приём очередного бита

Немного обобщим: если частота приёмника μ в N раз больше частоты протокола ν, то фронт с номером $N \cdot (i+1) + \lfloor \frac{N}{2} \rfloor$ —

- либо самый близкий к середине интервала передачи $x[i]$, если N нечётно,
- либо один из двух фронтов, самых близких к середине этого интервала, если N чётно

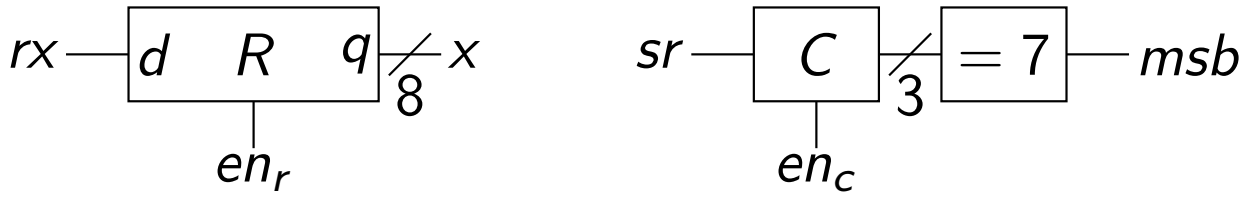
Окончательно обобщим: если μ не делится нацело на ν, то формула для номера среднего фронта бита $x[i]$ устроена сложнее, но если частота μ достаточно высокая, то можно считать, что она *приблизительно* в $\lfloor \frac{\mu}{\nu} \rfloor$ раз больше частоты ν, и отнести *реальное* расхождение в погрешность

Итог: бит $x[i]$ можно семплировать по фронту с номером

$$\lfloor \frac{\mu}{\nu} \rfloor \cdot (i+1) + \lfloor \frac{\mu}{2 \cdot \nu} \rfloor$$

UART: как реализовать приёмник

Операционный автомат



R — синхронный *сдвиговый регистр* ширины 8 с включением:

- на выход выдаётся сохранённое значение
- $en_r = 0 \Rightarrow$ сохранённое значение не изменяется
- $en_r = 1 \Rightarrow$ сохранённое значение сдвигается на один разряд в сторону младшего бита, и в старший бит записывается текущее значение d

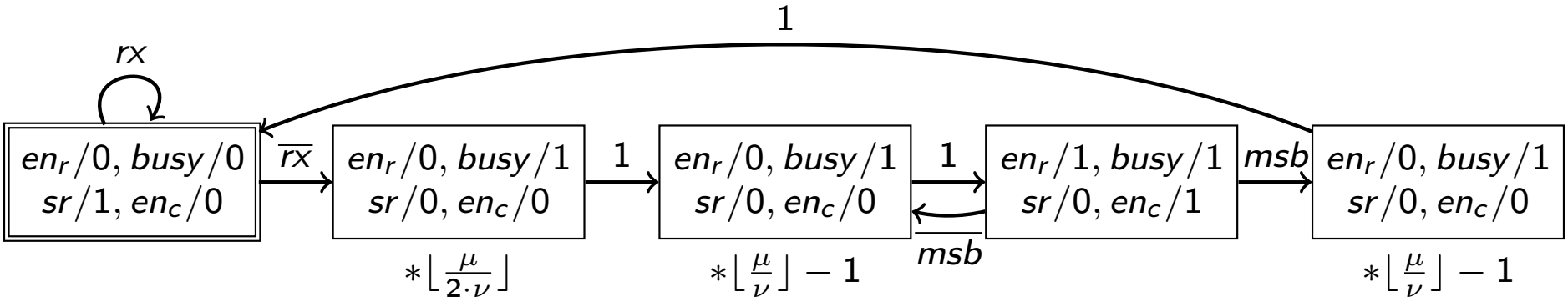
C — синхронный *счётчик* ширины 3, отсчитывающий число принятых разрядов данных:

- на выход выдаётся сохранённое значение
- $sr = 1 \Rightarrow$ сохраняется значеие 0
- $sr = 0$, $en_c = 1 \Rightarrow$ сохранённое значение увеличивается на 1

“$= 7$” — комбинационная схема, выход которой равен 1 $\Leftrightarrow$ вход равен 7

UART: как реализовать приёмник

Управляющий автомат

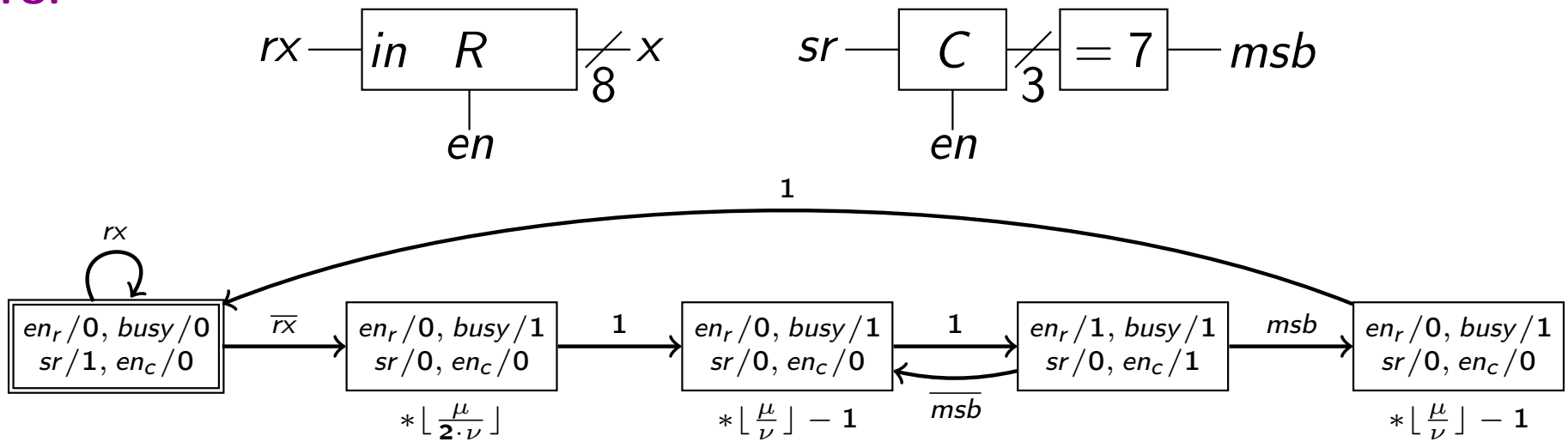


Состояния слева направо:

1. Тишина, приёма нет
2. Начался приём, ждём до среднего фронта начинающего бита
3. Ждём такта перед средним фронтом следующего бита: перед этим тактом выполняем переход
4. Следующий фронт — средний:
   - сдвигаем состояние R, увеличиваем счётчик C
   - переходим к приёму следующего бита (либо данных, либо завершающего)
5. Ждём до середины завершающего бита, чтобы избежать “ложного” начала приёма

UART: как реализовать приёмник

Итог



Эту схему можно легко адаптировать и к другим вариантам протокола:

- *Частота протокола:* учтена в числе $\frac{\mu}{\nu}$
- *Ширина пересылаемого значения:*
  поправить ширину R и C и число в подсхеме “$= M$”
- *Порядок пересылки разрядов:*
  подходящим образом соединить выход R с портом x
- *Механизмы обеспечения корректности:*
  добавить подсхему суммирования принятых битов,
  увеличить длительность правого состояния управляющего автомата