Языки описания схем

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

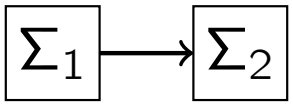
Блок K2

Кое-что ещё:
схемная реализация передатчика UART,
автоматы с таймерами

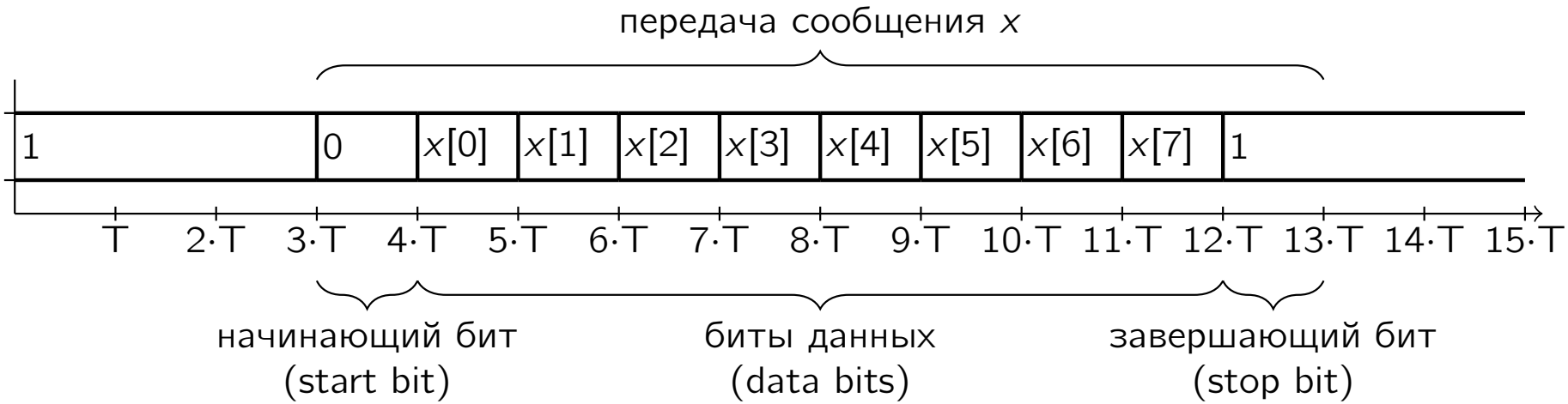
Лектор:
Подымов Владислав Васильевич
E-mail:
valdus@yandex.ru

ВМК МГУ, 2023/2024, 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 передатчика (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 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 с данными:

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

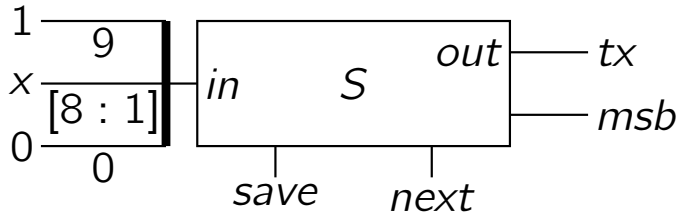
Схема, умеющая сохранять значение и последовательно выдавать его разряды, уже встречалась в практической части курса под названием «сериализатор»

UART: как реализовать передатчик

Операционный автомат

Используем в операционном автомате такой сериализатор S:

- Порты:
  - вход *in* ширины 10 (*все биты сообщения*)
  - входы *save* и *next* ширины 1
  - выходы *out*, *msb* ширины 1
- В сериализаторе хранятся значение v ширины 10 (сообщение) и значение i ширины 4 (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

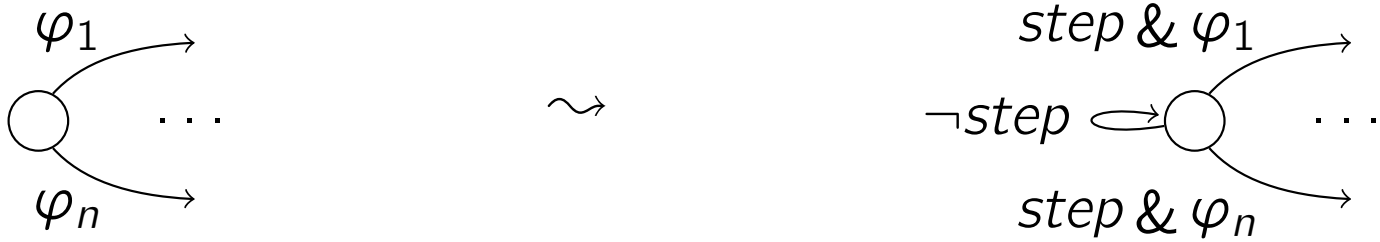


Автоматы с таймерами

Напоминание: в схеме, соответствующей автомату (как символьному, так и «обычному»), заданные значения выходных переменных (выходной символ) выдаются ровно один такт

Иногда при разработке автоматов возникает потребность ослабить это ограничение и заставить автомат оставаться в состоянии, выдавая один и тот же символ некоторое заранее заданное число тактов

Чтобы учесть эту потребность, можно разработать автомат без ограничения одного такта и слегка подправить его переходы при помощи добавочного управляющего сигнала:

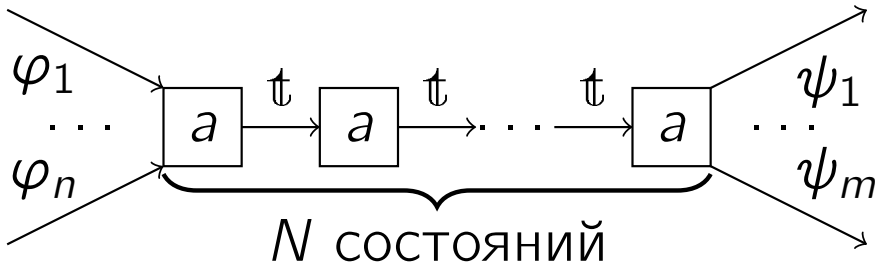


Типовая и нередко встречающаяся подсхема, реализующая такой сигнал — это таймер, отсчитывающий заданное число тактов

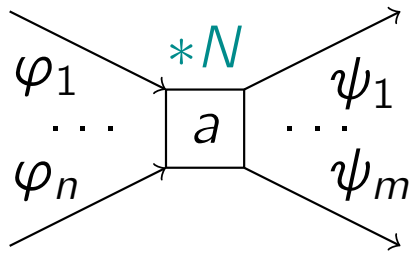
Поэтому имеет смысл обсудить, как может выглядеть типовая реализация символьного автомата, снабжённого таймерами для «растягивания» состояний на заданное число тактов

Автоматы с таймерами

Фрагмент автомата вида



будем изображать так:

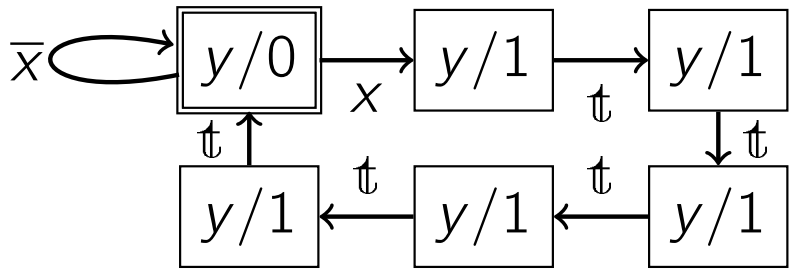


Такое сокращение можно понимать как состояние, в котором автомат должен находиться ровно N тактов вместо одного, после чего выполнение переходов продолжается обычным образом

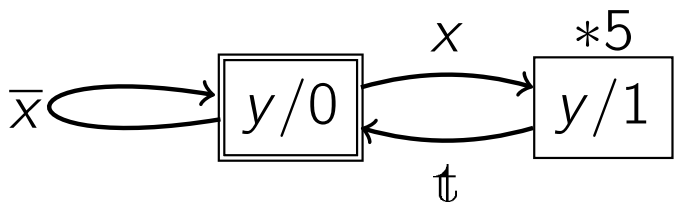
Для единообразия положим, что
по умолчанию каждому состоянию приписана метка «$*1$»

Автоматы с таймерами

Пример: автомат



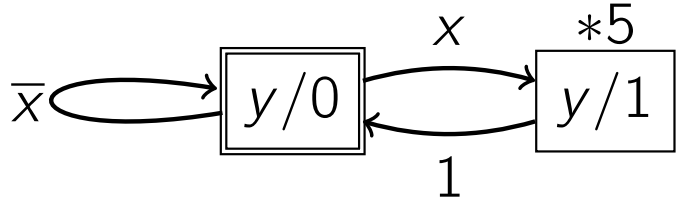
можно изобразить и так:



Нижнее изображение можно трактовать как автомат с двумя состояниями:

- начальное: в нём автомат находится 1 такт
- неначальное: в нём автомат находится 5 тактов

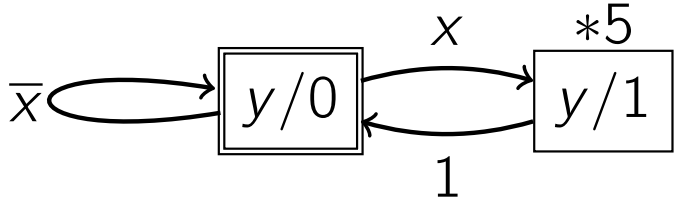
Автоматы с таймерами



Схемную реализацию такого доразмеченного автомата можно устроить так:

- Добавим в автомат один таймер, способный хранить чи́сла до $(N-1)$ включительно для наибольшего N среди всех «$*N$»
- При сбросе и выполнении явно изображённого перехода в состояние с меткой $*N$ в таймере сохраняется значение $(N-1)$
- Сохранённое значение уменьшается каждый такт, пока не достигнет ноля, и переходы автомата не выполняются
- Если в таймере сохранено значение 0 во время переднего фронта тактового сигнала, то выполняется следующий явно изображённый переход

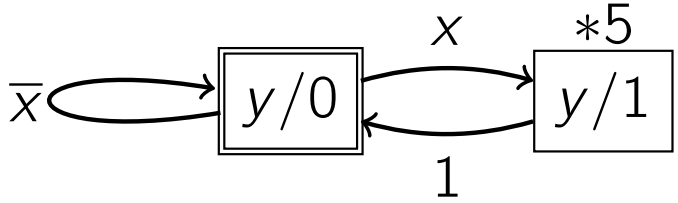
Автоматы с таймерами



Как это может выглядеть:

```
// Параметры и точки
localparam S_LEFT = 0, S_RIGHT = 1;
reg state; wire next_state;
reg [2:0] timer, next_timer; // Текущее значение таймера и грядущее значение (N-1)

// Регистр состояния и таймера
always @(posedge clk)
  if(rst) begin
    state <= S_LEFT;
    timer <= 0;                // Сброс ⇒ сохраняем (N-1) для начального состояния
  end else if(timer == 0) begin                // В таймере 0 ⇒ выполняем переход
    state <= next_state;
    timer <= next_timer;                         // Сохраняем новое значение (N-1)
  end else timer <= timer - 1;                 // В таймере не 0 ⇒ обратный отсчёт

// Функции выхода и переходов
assign y = (state == S_RIGHT);
assign next_state = ((state == S_LEFT && x) ? S_RIGHT : S_LEFT);

// Новое значение (N-1)
always @* begin
  next_timer = 0;                                      // По умолчанию (N-1) - это 0
  if(next_state == S_RIGHT) next_timer = 4;                 // (N-1) не по умолчанию
end
```

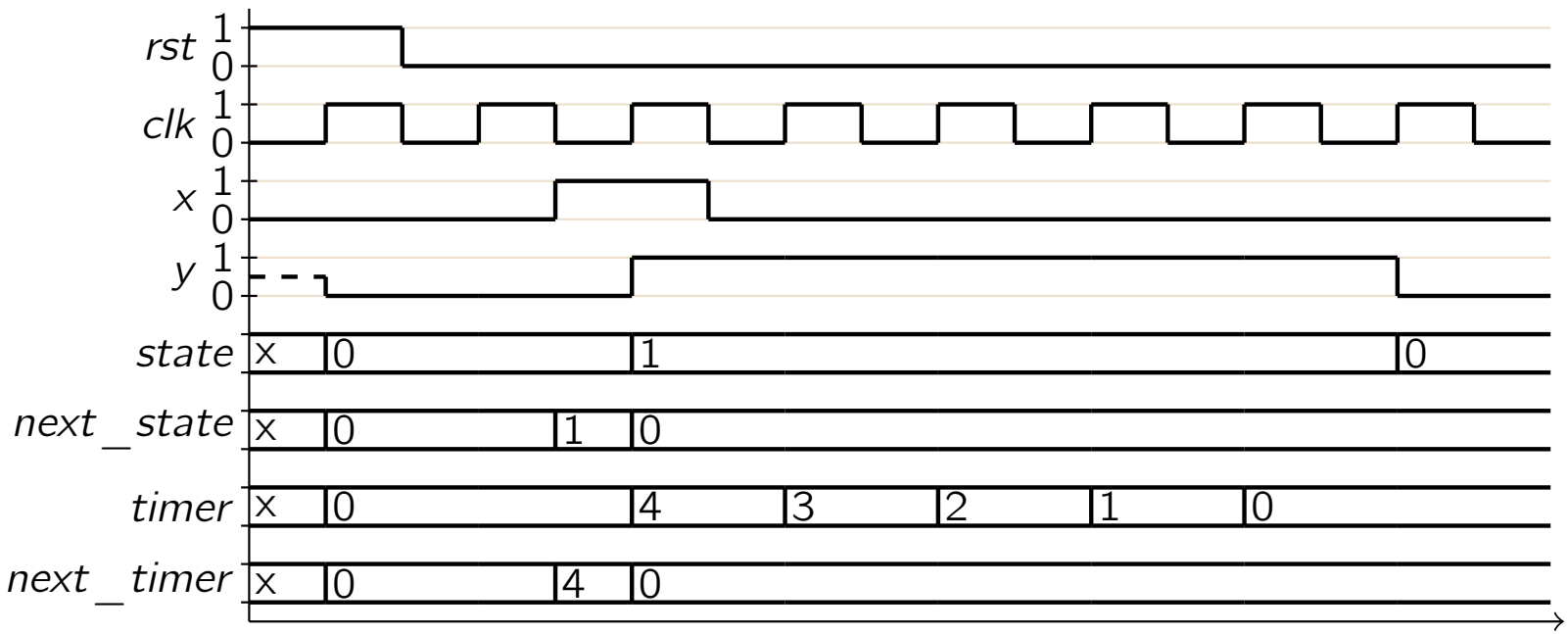
Автоматы с таймерами




```
// Регистр состояния и таймера
always @(posedge clk)
  if(rst) begin
    state <= S_LEFT;
    timer <= 0;
  end else if(timer == 0) begin
    state <= next_state;
    timer <= next_timer;
  end else timer <= timer - 1;
```

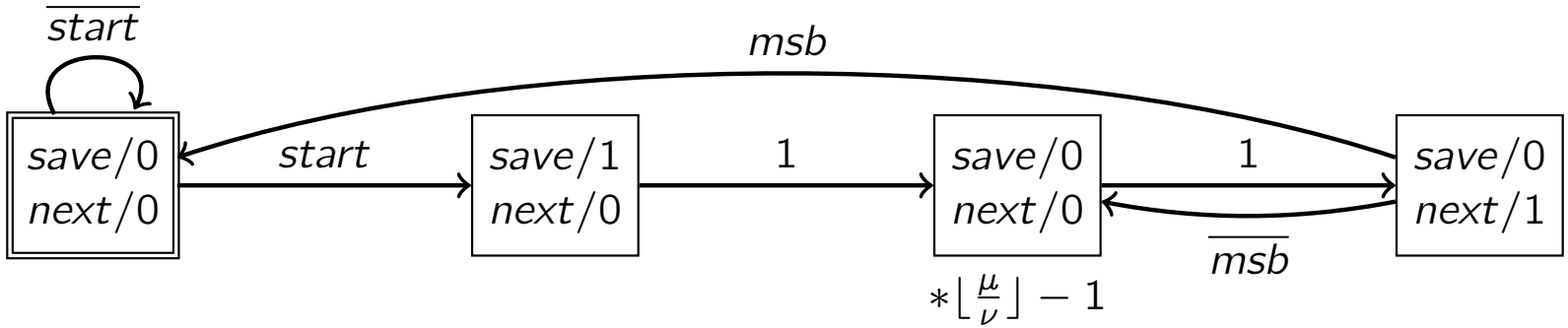
```
// Функции выхода и переходов
assign y = (state == S_RIGHT);
assign next_state = ((state == S_LEFT && x) ? S_RIGHT : S_LEFT);

// Новое значение (N-1)
always @* begin
  next_timer = 0;
  if(next_state == S_RIGHT) next_timer = 4;
end
```

Пример схемного выполнения такого автомата с таймером:



UART: как реализовать передатчик

Управляющий автомат:

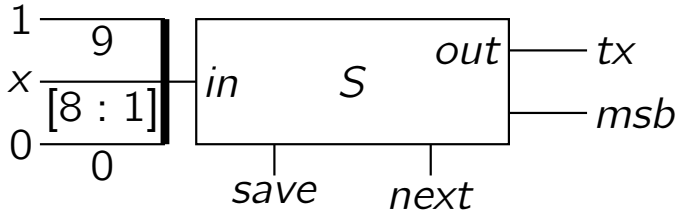


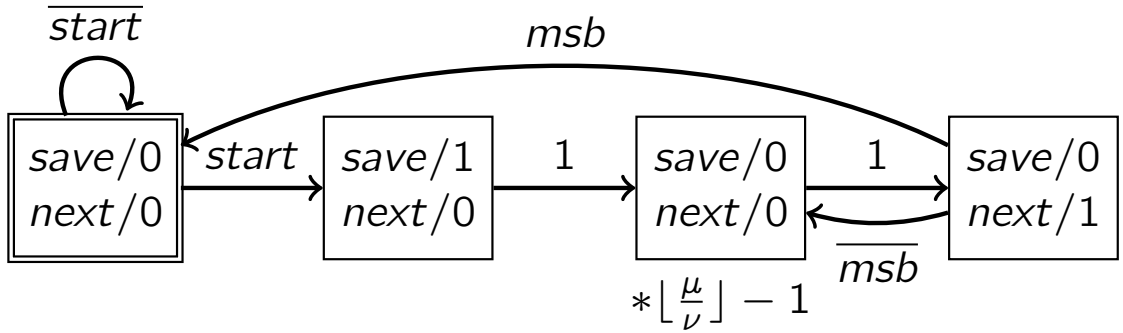
Состояния слева направо: (μ — частота передатчика)

1. Выдаём в *tx* тишину, не меняем состояние сериализатора
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*
- Проверка корректности передачи:
  расширить S требуемыми разрядами
  и соединить их с требуемыми значениями
  (1, сумма разрядов x, отрицание суммы разрядов x)