Языки описания схем

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

Блок 24

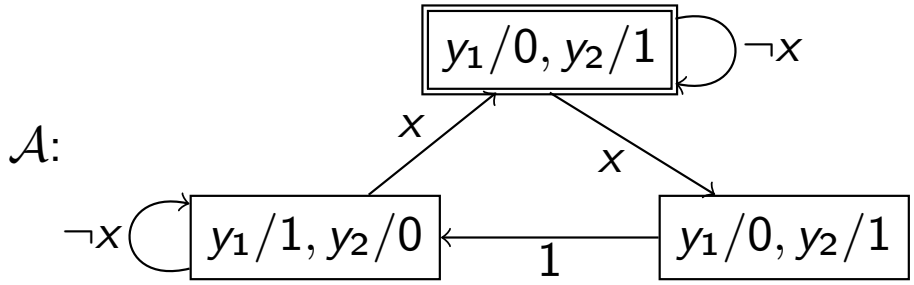
Как уменьшить автомат
Автоматы с таймерами

Лектор:
Подымов Владислав Васильевич

E-mail:
valdus@yandex.ru

ВМК МГУ, 2020, осенний семестр

Вступление/напоминание



Схемная трактовка символьного автомата устроена так:

- Сброс схемы = перевод автомата в начальное состояние
- Автомат находится в каждом состоянии **ровно один такт**, и выполняет переходы между **каждой** парой соседних тактов
- Для определения выполняемого перехода используются значения сигналов непосредственно перед фронтом тактового сигнала
- Значения выходных сигналов однозначно определяются текущим состоянием автомата: это значения, помечающие состояние

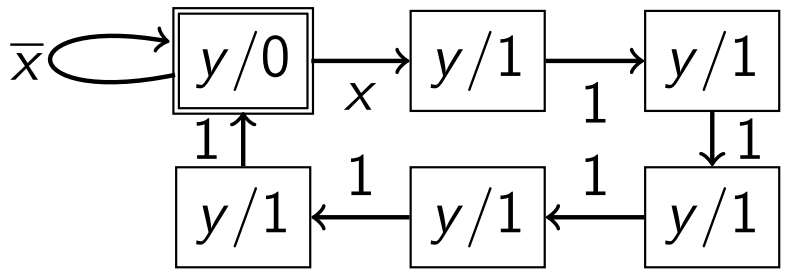
Из-за строгости такой трактовки *управляющие автоматы*, разрабатывающиеся типовым способом, могут выходить излишне громоздкими

Вступление/напоминание

Для примера рассмотрим такую синхронную схему со сбросом:

- x и y — соответственно вход и выход ширины 1
- $y(0) = 0$, и если $x(0) = x(1) = \ldots x(t) = 0$, то $y(t+1) = 0$
- Как только прочитано значение $x(t) = 1$, выдаются значения $y(t+1) = \cdots = y(t+5) = 1$, и затем схема сбрасывается

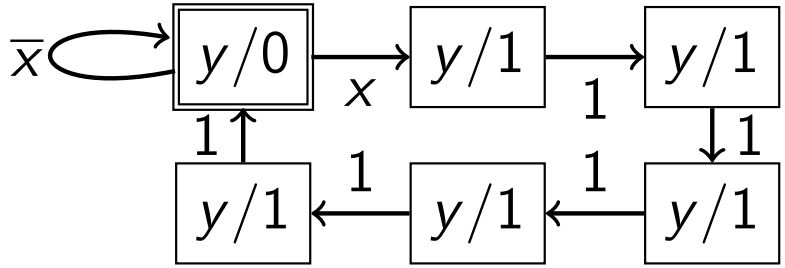
Если представить эту схему в виде символьного автомата ($\mathfrak{A}$),
то он будет выглядеть как-то так:



Состояния автомата $\mathfrak{A}$ можно "содержательно" разбить на два класса:
"выдаётся значение $y/0$" (начальное), и
"выдаётся значение $y/1$ (остальные)

Попробуем преобразовать $\mathfrak{A}$ так,
чтобы во втором классе изображалось не пять состояний, а одно

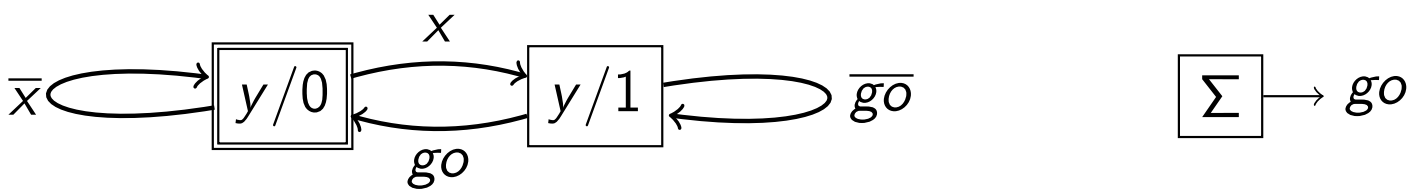
Слияние состояний автомата



Если объединить все пять состояний “$y/1$” в одно,
то из этого состояния должна быть возможность перейти

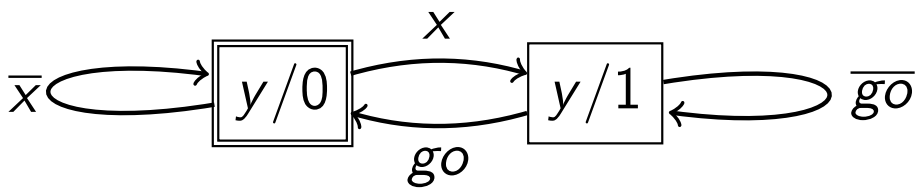
- в состояние “$y/0$” и
- в себя

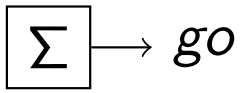
Детерминированный выбор одного из получившихся переходов можно обеспечить при помощи подходящей управляющей точки (go)
Для задания значения go понадобится отдельная подсхема (Σ) — *таймер*, отсчитывающий нужное число тактов



В общем случае Σ — это произвольная схема, реализующая требуемое условие выбора переходов автомата (*не обязательно таймер*)

Слияние состояний автомата





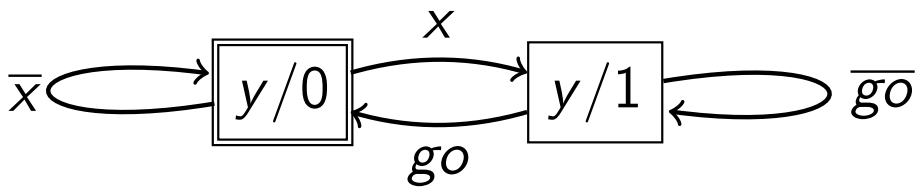
На языке Verilog это может выглядеть так:

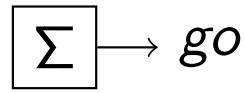
```
// Параметры и точки автомата и схемы Σ
localparam S_LEFT = 0, S_RIGHT = 1;
reg state; wire next_state;
reg [2:0] timer; wire go;

// Автомат
always @(posedge clk)                          // В этом примере сброс синхронный
  if(rst) state <= S_LEFT;
  else    state <= next_state;
assign y = (state == S_LEFT);
assign next_state = ((state == S_LEFT ? x : !go) ? S_RIGHT : S_LEFT);

// Схема Σ
always @(posedge clk)
  if(state == S_LEFT && next_state == S_RIGHT) // Если переход слева направо:
    timer <= 4;              // начинаем отсчёт дополнительных тактов ожидания.
  else if(next_state == S_RIGHT)                          // Если петля справа:
    timer <= timer - 1;                  // уменьшаем число тактов ожидания.
assign go = (timer == 0);   // Нет тактов ожидания ⇔ разрешён переход влево.
```

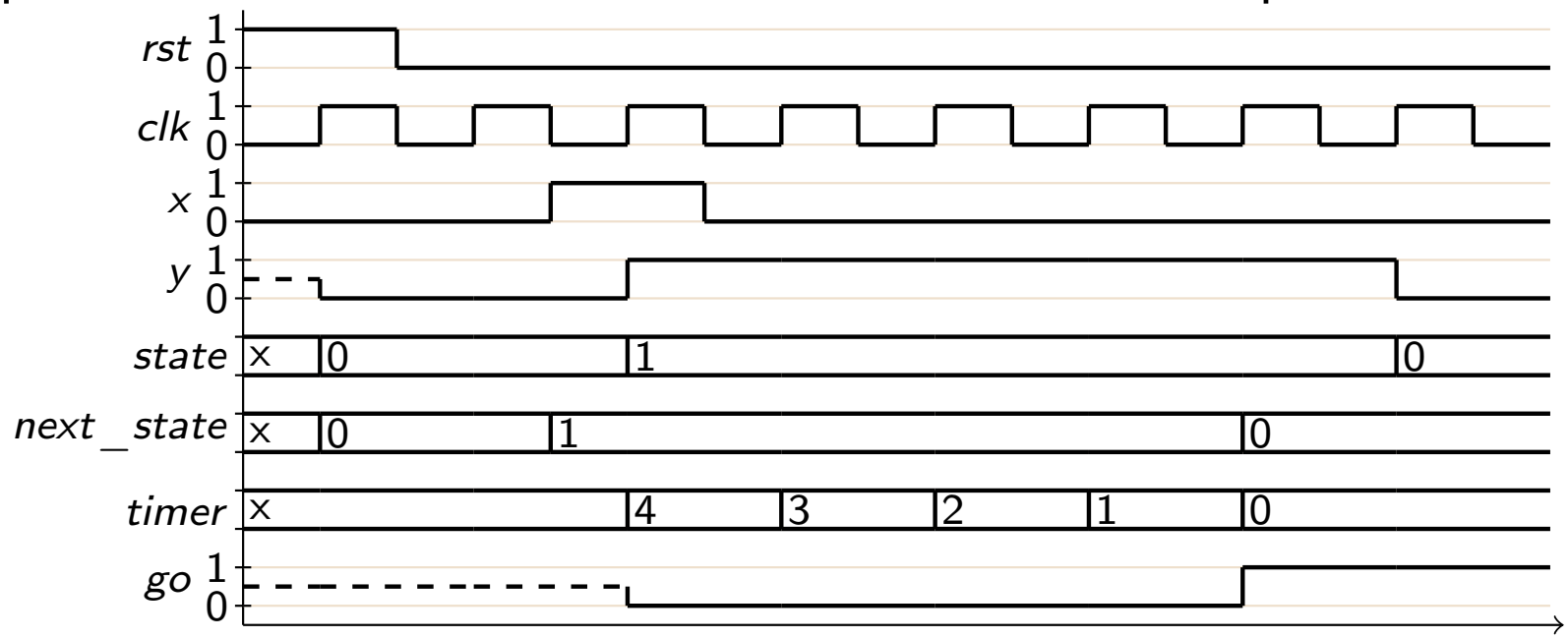
Слияние состояний автомата






```
// Схема Σ
always @(posedge clk)
  if(state == S_LEFT && next_state == S_RIGHT)
    timer <= 4;
  else if(next_state == S_RIGHT)
    timer <= timer - 1;
assign go = (timer == 0);
```

Пример схемного выполнения такого автомата с таймером:



Автоматы с таймерами

Необходимость приостановить выполнение переходов автомата
на заданное число тактов
нередко возникает при разработке управляющих схем

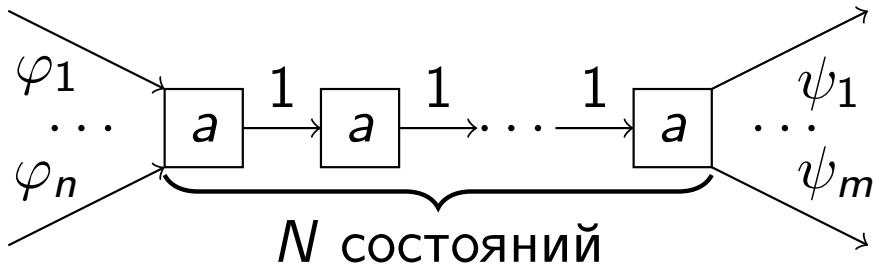
Такая приостановка может потребоваться для нескольких состояний,
и для каждого состояния — на своё число тактов

Так как автомат не может находиться
в нескольких состояниях одновременно,
для реализации такой приостановки
достаточно использовать один общий таймер

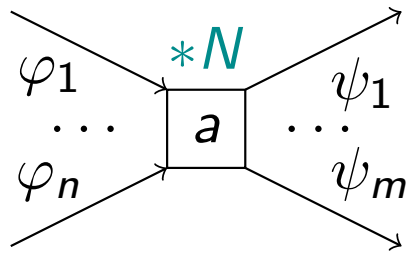
Обсудим удобочитаемое изображение и типовую реализацию
автомата с таким таймером

Автоматы с таймерами

Фрагмент автомата вида



будем изображать так:

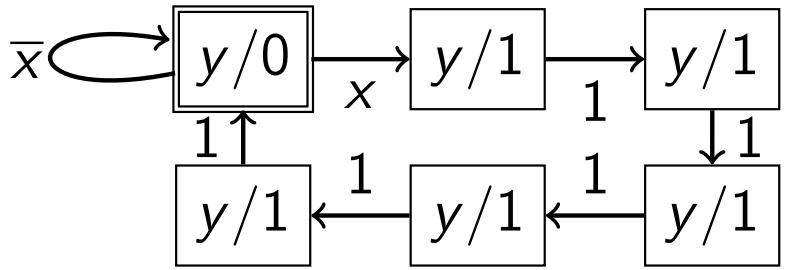


Если считать, что у автомата есть только **явно** изображённые состояния, то метка "$*N$" приобретает следующий смысл: автомат должен продержаться в состоянии **ровно N тактов**, после чего выполнить переход обычным образом

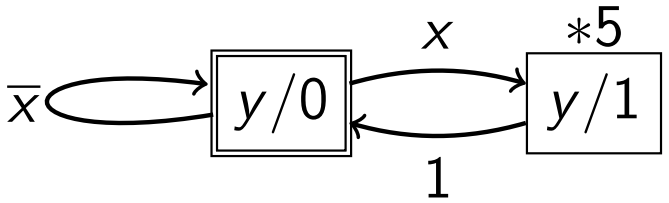
Для единообразия положим, что
по умолчанию каждому состоянию приписана метка "$*1$"

Автоматы с таймерами

Пример: автомат



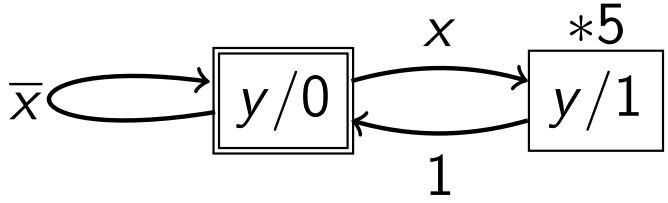
можно изобразить и так:



Нижнее изображение можно трактовать как автомат с **двумя** состояниями:

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

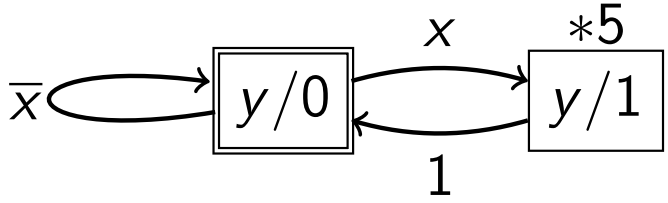
Автоматы с таймерами



Схемную реализацию такого доразмеченного автомата можно устроить так:

- Добавим в автомат один таймер, способный сохранить число $(N-1)$ для наибольшего N среди всех $*$-меток
- При сбросе и выполнении явно изображённого перехода в состояние с меткой $*N$ в таймере сохраняется значение $(N-1)$
- Если в таймере сохранено значение 0, то выполняется следующий явно изображённый переход, а иначе значение в таймере уменьшается на единицу (*и в автомате ничего не происходит*)

Автоматы с таймерами



Как это может выглядеть:

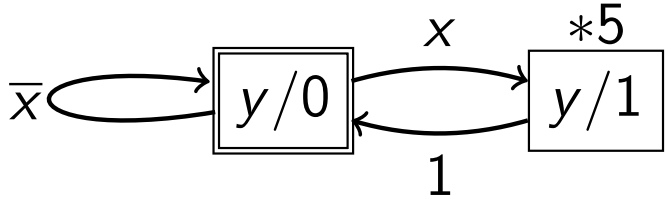
```
// Параметры и точки
localparam S_LEFT = 0, S_RIGHT = 1;
reg state; wire next_state;
reg [2:0] timer, next_timer; // Текущее значение таймера и грядущее значение (N-1)

// 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: