Языки описания схем

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

Блок 23

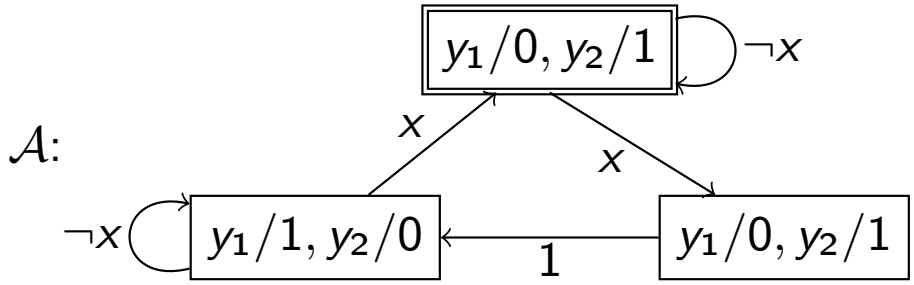
Автоматы с приостановкой выполнения переходов

Лектор:
Подымов Владислав Васильевич

E-mail:
valdus@yandex.ru

ВМК МГУ, 2019, 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

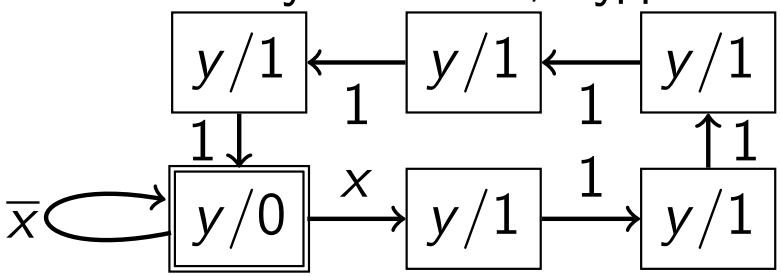
Из-за строгости такой трактовки *управляющие автоматы*, разрабатывающиеся типовым способом, могут выходить излишне громоздкими

Приостановка автомата

Рассмотрим такую синхронную схему со сбросом:

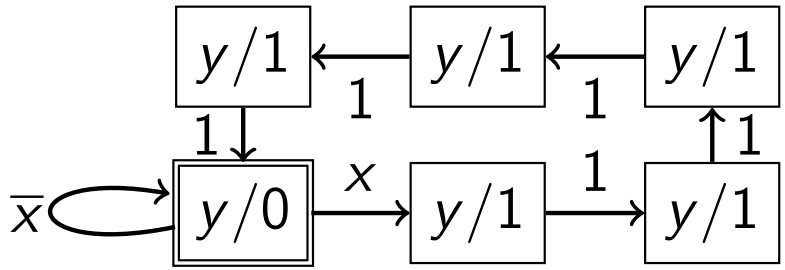
- x и y — соответственно вход и выход ширины 1
- После сброса и до прочтения значения $x = 1$ выдаётся значение $y = 0$
- После прочтения значения $x = 1$ **в течение пяти тактов** выдаётся значение $y = 1$, и затем схема сбрасывается

Если *строго и бездумно* придерживаться схемной трактовки автомата, то автомат, которому соответствует схема, будет выглядеть так:



Состояния автомата можно “содержательно” разбить на два класса:
“выдаётся значение $y/0$” (начальное), и
“выдаётся значение $y/1$ (остальные)
Преобразуем автомат так,
чтобы во втором классе оказалось не пять состояний, а одно

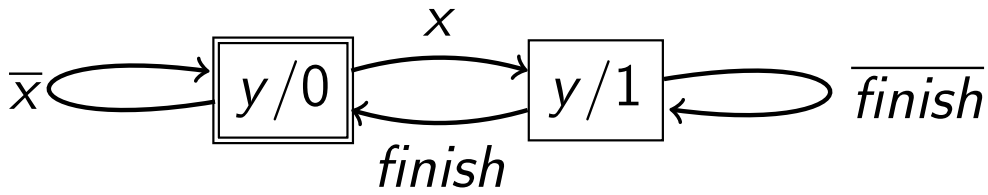
Приостановка автомата: локальный счётчик

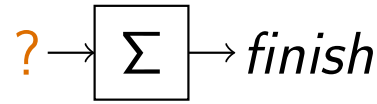


Цепочку состояний с одинаковыми выходами
можно заменить на одно состояние, например, так:

- Введём новый управляющий сигнал *finish*: индикатор конца цепочки
- Заменим цепочку состояний с безусловными переходами
  на одно состояние с переходами по значению *finish*
- Добавим подходящую подсхему управления сигналом *finish*

Приостановка автомата: локальный счётчик





Схему Σ можно устроить так:

- В ней реализован обратный отсчёт числа тактов, которые автомат должен находиться в состоянии “сверх положенного”
- Отсчёт начинается при переходе в состояние
- $finish = 1 \Leftrightarrow$ отсчёт завершён (число оставшихся “сверх”-тактов — 0)

На языке Verilog это может выглядеть так:

```
localparam S_LEFT = 0, S_RIGHT = 1;
reg state, next_state;
reg [2:0] counter; // здесь отсчитываем добавочные такты
wire finish; // индикатор конца цепочки состояний
// ... (типовая реализация автомата)
assign finish = (counter == 0);
always @(posedge clk)
  if(next_state == S_RIGHT) begin // нужно начать или продолжить отсчёт
    if(state == S_LEFT) counter <= 4; // начинаем отсчёт
    else counter <= counter - 1; // продолжаем отсчёт
  end else counter <= 1'bx; // отсчёта нет, значение счётчика неважно
```

Приостановка автомата

Необходимость “приостановить” автомат на заданное число тактов
нередко возникает при разработке управляющих схем

При этом в рамках одного автомата такая приостановка
может потребоваться для нескольких состояний,
и для каждого состояния — на своё число тактов

Расширим понятие автомата так, чтобы

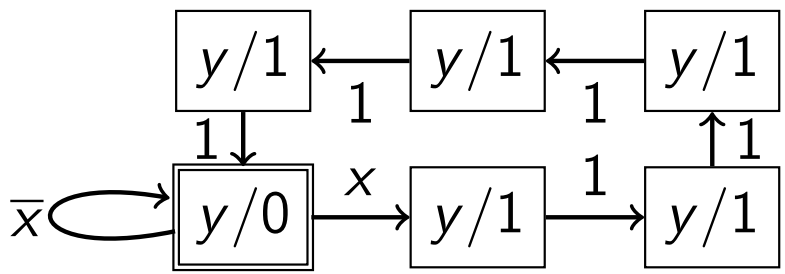
- наличие и разнообразие “приостановок”
  не влияло на наглядность автомата
- можно было предложить типовую реализацию
  расширенного автомата,
  не очень сильно отличающуюся от реализации *обычного* автомата

Приостановка автомата: глобальный счётчик

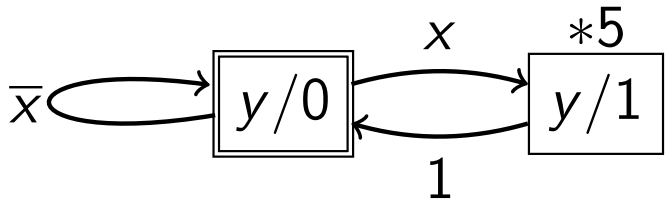
Расширим автомат так:

- Припишем каждому состоянию метку "$*N$":
  "автомат находится состоянии N тактов, и затем выполняет переход"
- Метки "$*1$" будем опускать

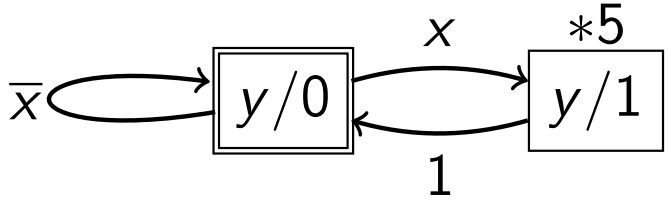
Пример



Эквивалентный расширенный автомат:



Приостановка автомата: глобальный счётчик

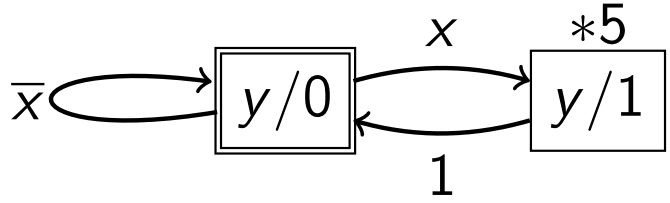


Особенности выполнения такого автомата:

- При переходе в **любое** состояние начинается обратный отсчёт
- Если состояние помечено записью “$*N$”,
  то отсчёт начинается с числа $(N-1)$
  - То есть текущее отсчитываемое число — это то,
    на сколько тактов следует **отложить** выполнение перехода
- Если отсчёт дошёл до числа 0,
  то при чтении следующего символа выполняется переход

Отразим эти особенности в типовой реализации автомата

Приостановка автомата: глобальный счётчик




```
localparam S_LEFT = 0, S_RIGHT = 1;
localparam C_LEFT = 0, C_RIGHT = 4; // (N-1) для каждого состояния
reg state, next_state;
reg [2:0] counter, next_counter;

always @(posedge clk, posedge rst)
  if(rst) begin
    state <= S_LEFT;
    counter <= C_LEFT; // в левом состоянии задерживаемся на (C_LEFT + 1) такт
  end else if(counter == 0) begin
    state <= next_state;
    counter <= next_counter; // если переходим в новое состояние, то начинаем отсчёт
  end else counter <= counter - 1; // если не переходим в новое состояние, то продолжаем отсчёт

assign y = (state == S_RIGHT); // функция выхода -- как обычно

always @* begin
  next_state = S_LEFT; // по умолчанию переходим влево ...
  next_counter = C_LEFT; // ... и остаёмся в нём один такт: C_LEFT + 1
  if(state == S_LEFT && x) begin
    next_state = S_RIGHT; // не по умолчанию переходим вправо ...
    next_counter = C_RIGHT; // ... и остаёмся в нём 5 тактов: C_RIGHT + 1
  end
end
```