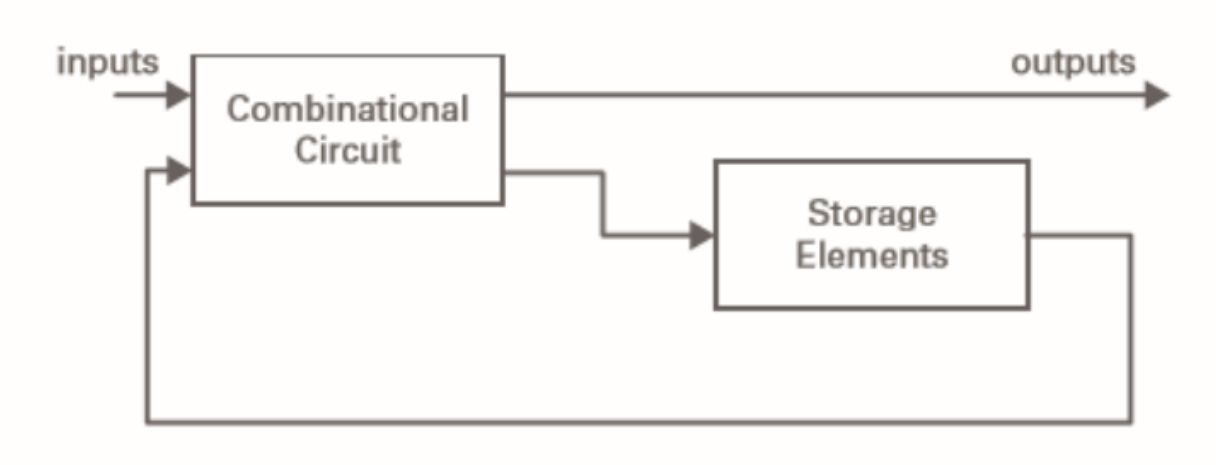
Конечные автоматы (КА)

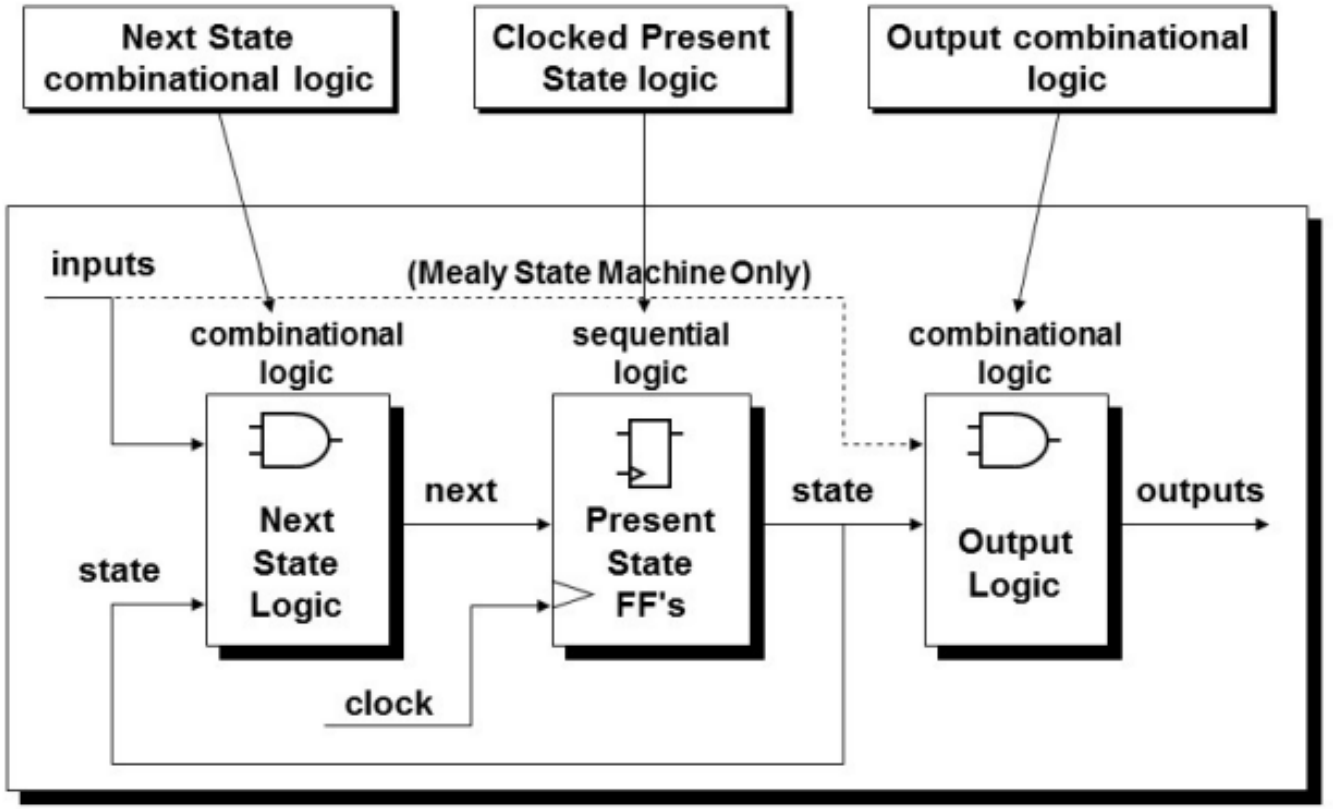
- Математическая модель последовательностных схем
- Используются для реализации логики управления функциональных блоков



Конечные автоматы (КА)

- Автомат Мура – выходные сигналы зависят только от текущего состояния
- Автомат Мили – выходные сигналы зависят от текущего состояния и входных сигналов

Структурная схема КА



Конечные автоматы (КА)

- Работа автомата синхронизируется по тактовому сигналу
- Текущее состояние автомата хранится в регистре
- Разрядность регистра определяется количеством состояний
- Логика формирования следующего состояния – комбинационная схема
- Логика формирования выходных сигналов – комбинационная схема
- Различаются по способу кодирования состояний (двоичное и прямое кодирование)

Двоичные КА

- Каждому состоянию ставится в соответствие двоичное число
- Для КА с N состояниями нужно ceil($\log_2 N$) триггеров для представления состояния

Прямое кодирование

- Для каждого состояния используется один бит (в каждый момент времени только в одном из разрядов содержится единица)
- Для КА с N состояниями нужно N триггеров для представления состояния:
  - Требует большего количества триггеров, чем двоичное
  - Логика формирования следующего состояния и формирования выходных сигналов проще

“Хороший” КА

- Легко модифицируется (позволяет легко добавлять новые состояния, измененять входные и выходные сигналы)
- Прост для понимания
- Лаконичный (больше кода = больше ошибок)

Описание в виде графа

- Могут быть представлены в виде графа
- Вершины – состояния
- Ребра – переходы между состояниями
- Значения над ребрами – входные сигналы, вызвавшие переход
- Для автомата Мили над ребрами указываются не только входные сигналы, но и соответствующие значения выходных сигналов

Пример задачи, решаемой при помощи конечного автомата

- Разработать схему поиска последовательности “01” на входе
- При нахождении последовательности выдавать на выход “1”
- В остальных случаях выдавать “0”

Пример автомата Мура

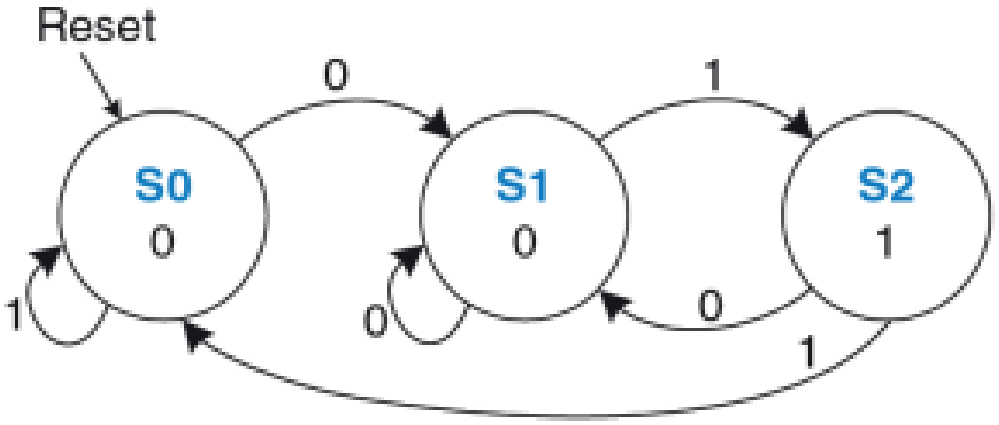


Таблица выходов

Текущее состояние	Выход
S0	0
S1	0
S2	1

Таблица переходов

Текущее состояние	Вход	Следующее состояние
S0	0	S1
S0	1	S0
S1	0	S1
S1	1	S2
S2	0	S1
S2	1	S0

Пример автомати Мили

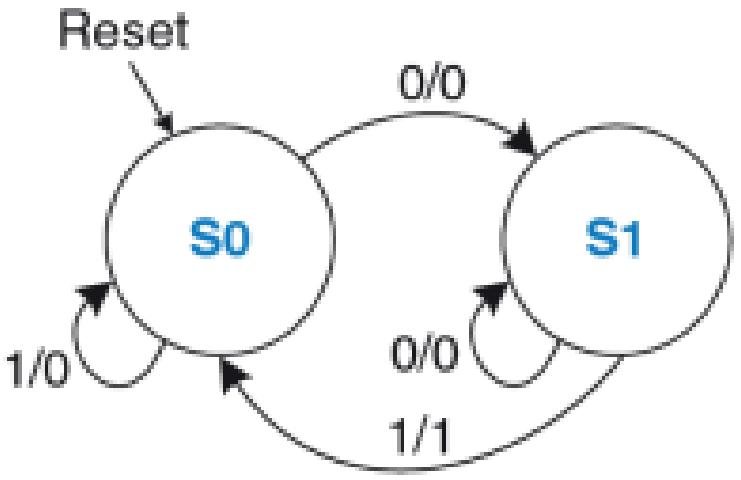


Таблица переходов и выходов

Текущее состояние	Вход	Следующее состояние	Выход
S0	0	S1	0
S0	1	S0	0
S1	0	S1	0
S3	1	S0	1

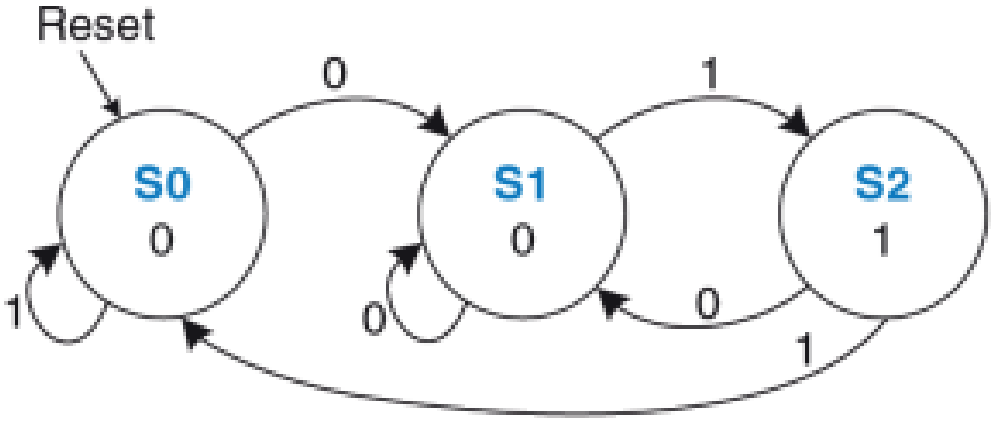
Описание на SV

- Комбинационная схема, реализующая логику переходов (always_comb, assign)
- Синхронная логика для смены состояний (always_ff)
- Комбинационная схема, описывающая логику формирования выходных сигналов (always_comb, assign)

- Для описания состояний КА часто используется тип enum
- enum – это тип данных, который присваивает имена целочисленным константам
- Такое представление состояний упрощает код и облегчает тестирование КА (в симуляции регистр состояний будет содержать символьные имена состояний)

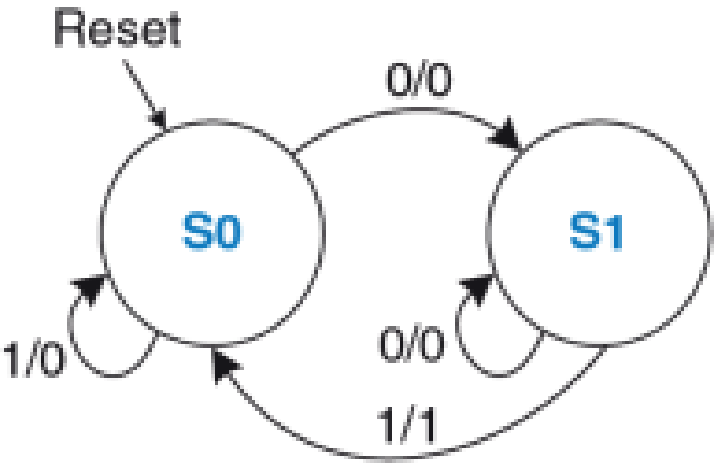
```
enum logic [1:0] {S0 = 2'b00, S1 = 2'b01, S2 = 2'b10}  state, next;

always_ff @(posedge clk) begin
    if (rst)    state <= S0;
    else        state <= next;
end

always_comb begin
    next = state;
    case (state)
        S0 : if (in == 0) next = S1;
        S1 : if (in == 1) next = S2;
        S2 :
        begin
            if(in == 1)    next = S0;
            else           next = S1;
        end
        default :          next = S0;
    endcase
end

assign out = (state == S2);
```

Пример автомата Мура

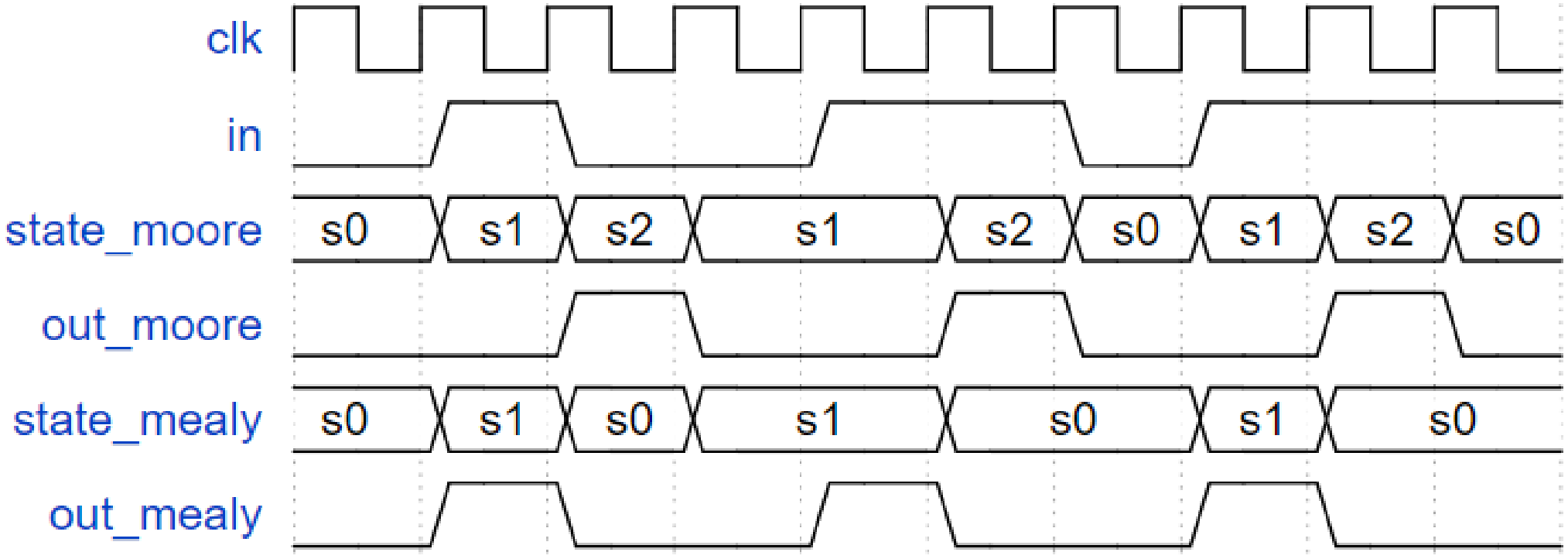


```
enum logic [1:0] {S0 = 2'b00, S1 = 2'b01}  state, next;

always_ff @(posedge clk) begin
    if (rst) state <= S0;
    else     state <= next;
end

always_comb begin
    next = state;
    case (state)
        S0 : if (in == 0) next = S1;
        S1 : if (in == 1) next = S0;
        default :         next = S0;
    endcase
end

assign out = (state == S1) & in;
```

Пример автомата Мили



Временные диаграммы автоматов Мили и Мура



Литература

- Дэвид М. Хэррис, Сара Л. Хэррис: “Цифровая схемотехника и архитектура компьютера”
- IEEE Std 1800-2017 Standard for SystemVerilog - Unified Hardware Design, Specification, and Verification Language
- Clifford E. Cummings, Heath Chambers: “Finite State Machine (FSM) Design & Synthesis using SystemVerilog - Part I”