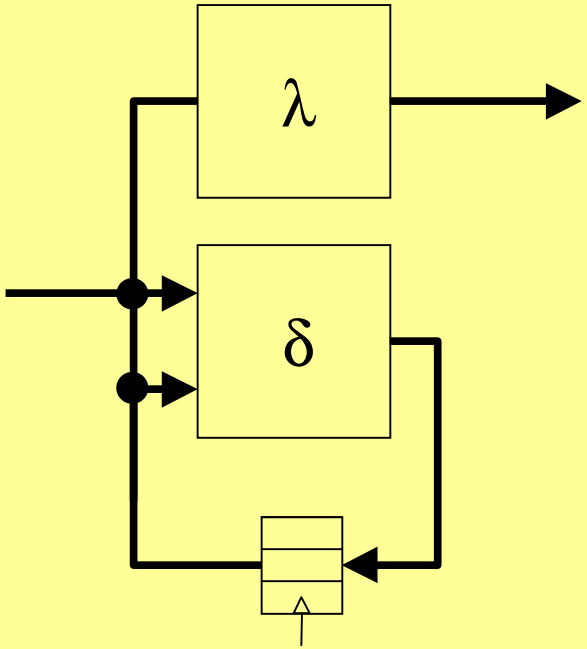
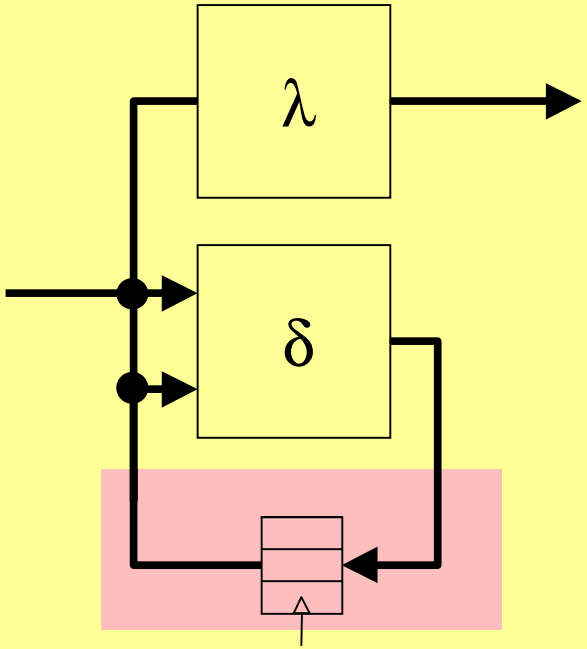
Endliche Automaten in Verilog

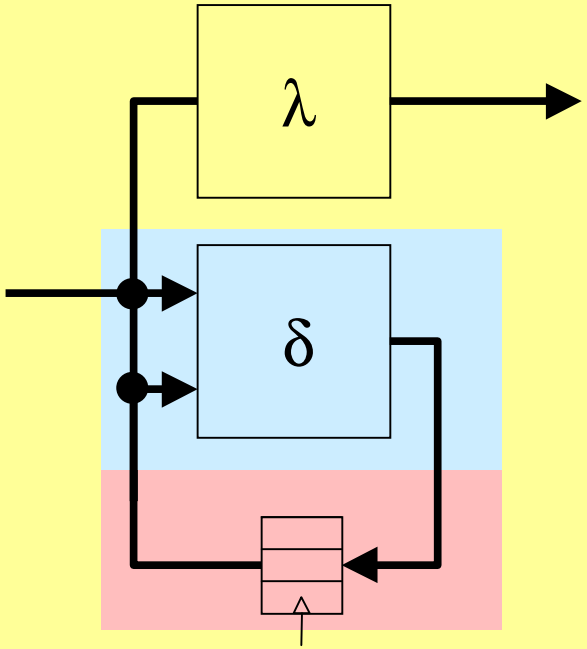


Endliche Automaten in Verilog



Register (Zustandsspeicher)

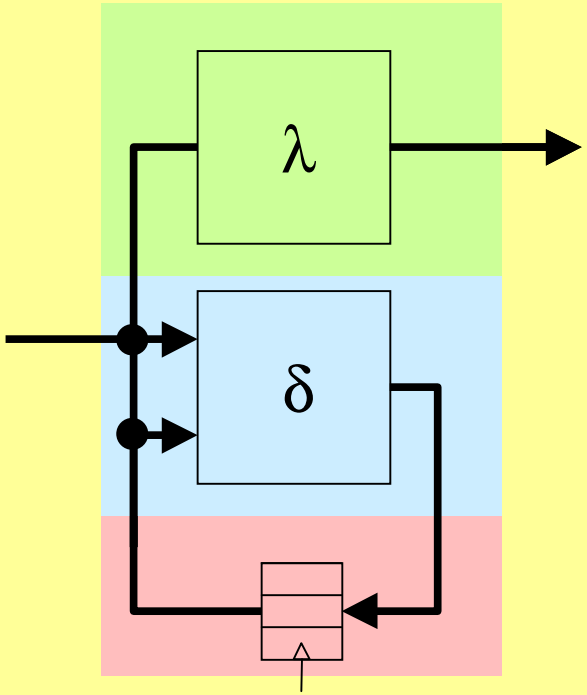
Endliche Automaten in Verilog



Übergangsfunktion

Register (Zustandsspeicher)

Endliche Automaten in Verilog



Ausgangsfunktion

Übergangsfunktion

Register (Zustandsspeicher)

Kombinationsmöglichkeiten

Zustandsregister	Übergangsfunktion	Ausgangsfunktion	Typ
separat	separat	separat	1
kombiniert	kombiniert	separat	2
separat	kombiniert	kombiniert	3
kombiniert	separat	kombiniert	4
kombiniert	kombiniert	kombiniert	5

FSMs in Verilog: Typ1

- Jede Einheit in einen separaten always-Block:

```
always @(posedge clk)  state = next_state

always @(state or inp1 or ... or inpn) begin
  next_state = state;
  case (state)
            ...
  endcase

always @(state or inp1 or ... or inpn) begin
    ...
end
```

FSMs in Verilog: Typ1

- Vorteile:
  - übersichtlich
  - einfache Wartbarkeit
  - MOORE and MEALY-Maschinen
- Nachteil
  - längste Darstellung
  - Übergangsfunktion wird bei jeder Eingabe aktualisiert

FSMs in Verilog: Typ2

Register + Übergangsfunktion

- Vorteile
  - kompakter als Typ1
  - effizienter, da Übergangsfunktion nur bei clk aktualisiert wird
  - MOORE and MEALY-Maschinen
- Nachteile
  - nicht so modular wie Typ1

FSMs in Verilog: Typ3

Ausgangsfunktion + Übergangsfunktion

- Vorteile
  - MOORE and MEALY-Maschinen
  - kompakter als Typ1
- Nachteile
  - sehr unübersichtlich
  - Übergangsfunktion wird bei jeder Eingabe aktualisiert-

FSMs in Verilog: Typ4

Register + Ausgangsfunktion

- Vorteile
  - kompakter als Typ1
- Nachteile
  - nur MOORE-Maschinen
  - Übergangsfunktion wird bei jeder Eingabe aktualisiert

FSMs in Verilog: Typ5

Register + Über- und Ausgangsfunktion

- Vorteile
  - kompakter als Typ1 (keine next_state-Variable nötig)
  - Übergangs- und Ausgangsfunktion nur bei clk aktualisiert
- Nachteile
  - nur MOORE-Maschinen

Zustandskodierung

- Boole‘sche Kodierung

```
`define IDLE      3'b00
`define CHOOSE    3'b01
`define WATER     3'b10
`define ORANGE    3'b11
```

- Gray Kodierung

```
`define IDLE      3'b00
`define CHOOSE    3'b01
`define WATER     3'b11
`define ORANGE    3'b10
```

Zustandskodierung

- One-Hot

```
`define IDLE      4'b0001
`define CHOOSE    4'b0010
`define WATER     4'b0100
`define ORANGE    4'b1000
```

- Bei MOORE-Maschinen: Ausgänge als Zustandskodierung

```
`define IDLE      4'b000  // (d=0, w=0, o=0)
`define CHOOSE    4'b100  // (d=1, w=0, o=0)
`define WATER     4'b110  // (d=1, w=1, o=0)
`define ORANGE    4'b101  // (d=1, w=0, o=1)
```