

Digitaltechnik

Wintersemester 2021/2022

12. Vorlesung












TECHNISCHE
UNIVERSITÄT
DARMSTADT





Umfrage zur letzten Woche

1. Einleitung
2. SystemVerilog für Zustandsautomaten
3. SystemVerilog Abschluss und Ausblick
4. Sequentielle Grundelemente
5. Speicherfelder
6. Zusammenfassung

Anwendungs- software		Programme
Betriebs- systeme		Gerätetreiber
Architektur		Befehle Register
Mikro- architektur		Datenpfade Steuerung
Logik		Addierer Speicher
Digital- schaltungen		UND Gatter Inverter
Analog- schaltungen		Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen

Agenda



TECHNISCHE
UNIVERSITÄT
DARMSTADT

1. Einleitung










2. SystemVerilog für Zustandsautomaten

3. SystemVerilog Abschluss und Ausblick

4. Sequentielle Grundelemente

5. Speicherfelder

6. Zusammenfassung

Anwendungs- software		Programme
Betriebs- systeme		Gerätetreiber
Architektur		Befehle Register
Mikro- architektur		Datenpfade Steuerung
Logik		Addierer Speicher
Digital- schaltungen		UND Gatter Inverter
Analog- schaltungen		Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen

Überblick der heutigen Vorlesung



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- ▶ SystemVerilog für Zustandsautomaten
- ▶ SystemVerilog Abschluss und Ausblick
- ▶ Sequentielle Grundelemente
- ▶ Speicherfelder



Harris 2013/2016
Kap. 4.6, 5.4, 5.5

Agenda



TECHNISCHE
UNIVERSITÄT
DARMSTADT

1. Einleitung










2. SystemVerilog für Zustandsautomaten

3. SystemVerilog Abschluss und Ausblick

4. Sequentielle Grundelemente

5. Speicherfelder

6. Zusammenfassung

Anwendungs- software		Programme
Betriebs- systeme		Gerätetreiber
Architektur		Befehle Register
Mikro- architektur		Datenpfade Steuerung
Logik		Addierer Speicher
Digital- schaltungen		UND Gatter Inverter
Analog- schaltungen		Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen



- ▶ Auffrischung FSMs: Vorlesung 9, Harris Kap. 3.4 und **LQ10-7**

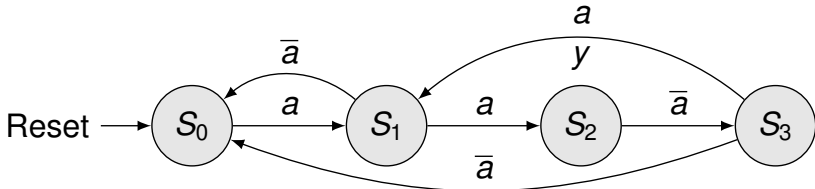
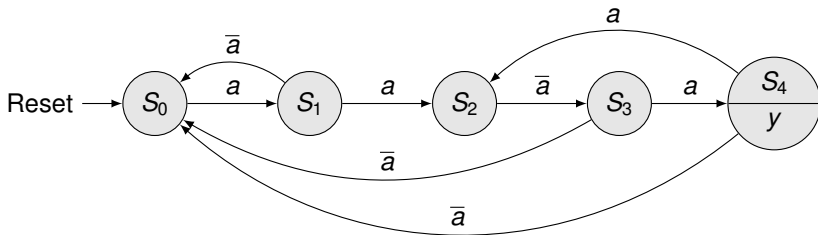
Vorgehen:

- ▶ Logikvektor oder `enum` für Zustände
- ▶ rücksetzbare Flip-Flops als Zustandsspeicher
- ▶ kombinatorische next-state Logik durch `case` in `always_comb` Block
- ▶ kombinatorische Ausgabe-Logik durch nebenläufige Zuweisungen

Moore- und Mealy-Automat für 1101 Mustererkennung (aus VL9)



TECHNISCHE
UNIVERSITÄT
DARMSTADT



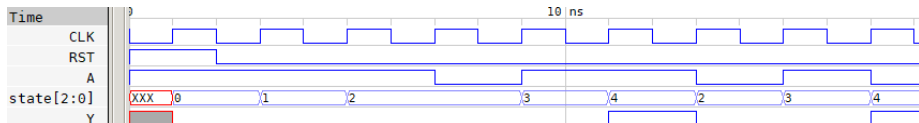
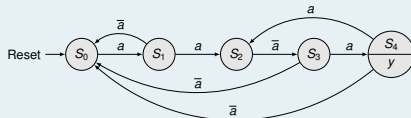
Moore Automat für 1101 Mustererkennung



TECHNISCHE
UNIVERSITÄT
DARMSTADT

fsm/pattern/moore.sv

```
1 module moore (input logic CLK, RST, A, output logic Y);
2   typedef enum logic [2:0] {S0, S1, S2, S3, S4} statetype;
3   statetype state, nextstate;
4   always_ff @(posedge CLK) state <= RST ? S0 : nextstate;
5   // next state logic
6   always_comb case (state)
7     S0:    nextstate = A ? S1 : S0;
8     S1:    nextstate = A ? S2 : S0;
9     S2:    nextstate = A ? S2 : S3;
10    S3:    nextstate = A ? S4 : S0;
11    S4:    nextstate = A ? S2 : S0;
12    default: nextstate = S0;
13  endcase
14  // output logic
15  assign Y = (state == S4);
16 endmodule
```



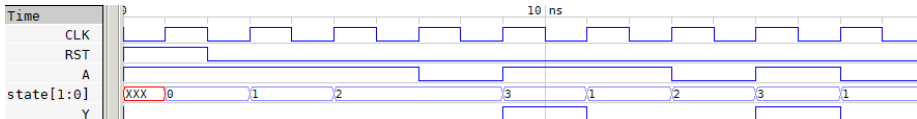
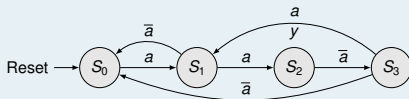
Mealy Automat für 1101 Mustererkennung



TECHNISCHE
UNIVERSITÄT
DARMSTADT

fsm/pattern/mealy.sv

```
1 module mealy (input logic CLK, RST, A, output logic Y);
2   typedef enum logic [1:0] {S0, S1, S2, S3} statetype;
3   statetype state, nextstate;
4   always_ff @(posedge CLK) state <= RST ? S0 : nextstate;
5   // next state logic
6   always_comb case (state)
7     S0:      nextstate = A ? S1 : S0;
8     S1:      nextstate = A ? S2 : S0;
9     S2:      nextstate = A ? S2 : S3;
10    S3:      nextstate = A ? S1 : S0;
11    default: nextstate = S0;
12  endcase
13  // output logic
14  assign Y = (state == S3 && A);
15 endmodule
```



Agenda



TECHNISCHE
UNIVERSITÄT
DARMSTADT

1. Einleitung
2. SystemVerilog für Zustandsautomaten
3. SystemVerilog Abschluss und Ausblick
4. Sequentielle Grundelemente
5. Speicherfelder
6. Zusammenfassung

Anwendungs- software		Programme
Betriebs- systeme		Gerätetreiber
Architektur		Befehle Register
Mikro- architektur		Datenpfade Steuerung
Logik		Addierer Speicher
Digital- schaltungen		UND Gatter Inverter
Analog- schaltungen		Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen



- ▶ alle SystemVerilog Konstrukte sind grundsätzlich simulierbar
- ▶ aber nicht alle Simulatoren unterstützen den kompletten Sprachstandard
- ▶ nicht synthetisierbar sind
 - ▶ Signalinitialisierung bei der Deklaration
 - ▶ `initial` Blöcke
 - ▶ explizite Verzögerungen (per #)
 - ▶ die meisten Funktionen wie `$display`, `$time`, `$clog2` (ceiling of \log_2)
 - ▶ `real` Signale

- ▶ SystemVerilog ist Weiterentwicklung von Verilog (für Verifikation)
- ▶ Verilog immer noch weiter verbreitet
- ▶ im Rahmen der Veranstaltung nur wenige Unterschiede zu Verilog:
 - ▶ Verilog hat separate Datentypen statt `logic`
 - ▶ `wire` für Zuweisungen per `assign`
 - ▶ `reg` für Zuweisungen in `always` Blöcken
 - ▶ Verilog hat keine spezifischen `always` Blöcke für
 - ▶ Flip-Flops (`always_ff`): `always @(posedge clk)`
 - ▶ Latches (`always_latch`): `always @(clk, d)`
 - ▶ kombinatorische Logik (`always_comb`): `always @*`

⇒ i.d.R. ist SystemVerilog leichter verständlich

- ▶ Viele Sprachkonstrukte können in kurzer Einführung nicht behandelt werden

- ▶ Tasks, Funktionen und Programme
- ▶ Klassen und Vererbung
- ▶ Verifikationsunterstützung
- ▶ fork und join
- ▶ Events
- ▶ Präprozessor
- ▶ ...

⇒ bei tieferem Interesse weitere Literatur verwenden



WWW



Pause & Umfrage bis hier

Wiederholung: Schichtenmodell eines Computers



TECHNISCHE
UNIVERSITÄT
DARMSTADT










Anwendungs- software		Programme
Betriebs- systeme		Gerätetreiber
Architektur		Befehle Register
Mikro- architektur		Datenpfade Steuerung
Logik		Addierer Speicher
Digital- schaltungen		UND Gatter Inverter
Analog- schaltungen		Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen

Agenda

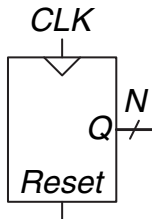


TECHNISCHE
UNIVERSITÄT
DARMSTADT

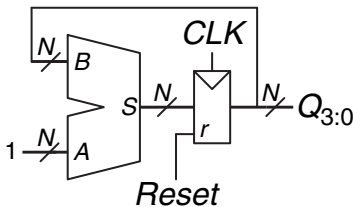
1. Einleitung
2. SystemVerilog für Zustandsautomaten
3. SystemVerilog Abschluss und Ausblick
4. Sequentielle Grundelemente
5. Speicherfelder
6. Zusammenfassung

Anwendungs- software		Programme
Betriebs- systeme		Gerätetreiber
Architektur		Befehle Register
Mikro- architektur		Datenpfade Steuerung
Logik		Addierer Speicher
Digital- schaltungen		UND Gatter Inverter
Analog- schaltungen		Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen

- ▶ Erhöht sich bei jeder steigenden Taktflanke
- ▶ Dient zum Durchlaufen von Zahlen. Zum Beispiel:
 - ▶ 000, 001, 010, 011, 100, 101, 110, 111, 000, 001...
- ▶ Verwendung (Beispiele):
 - ▶ Displays von Digitaluhren
 - ▶ Programmzähler: verfolgt die Befehlsausführung in einer CPU

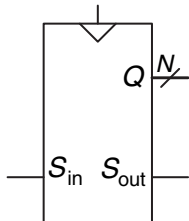


Symbol

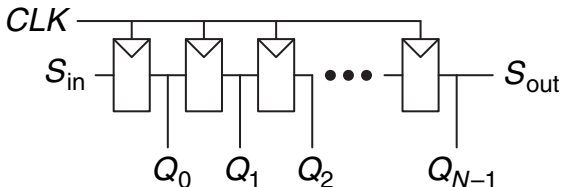


Implementierung

- ▶ Bei jeder steigenden Taktflanke wird der Speicherinhalt ein Flip-Flop weiter verschoben (FIFO-Prinzip: First In – First Out)
 - ▶ Neues Bit S_{in} wird eingelesen
 - ▶ Letztes Bit S_{out} wird nach außen verschoben/verworfen
- ▶ Seriell-Parallel-Wandler: Wandelt den seriellen Eingang (S_{in}) in den parallelen Ausgang ($Q_{0:N-1}$) um



Symbol



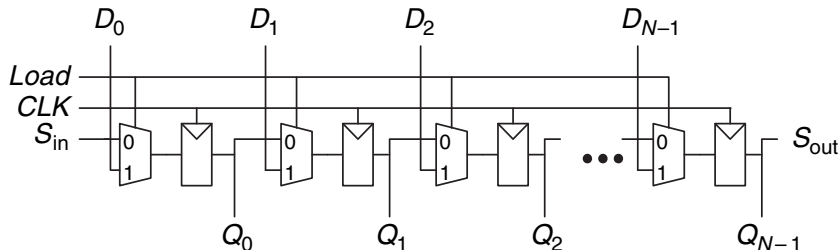
Implementierung

Schieberegister mit parallelem Laden



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- ▶ Für $Load = 1$: normales N -Bit Register
- ▶ Für $Load = 0$: Schieberegister
- ▶ Kann dadurch sowohl als Seriell-Parallel-Wandler (S_{in} zu $Q_{0:N-1}$, $Load = 0$) als auch als Parallel-Seriell-Wandler ($D_{0:N-1}$ zu S_{out} , $Load = 1$) fungieren












Agenda

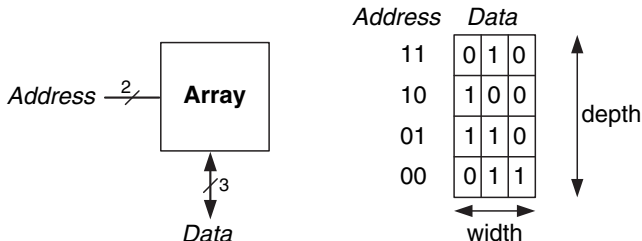
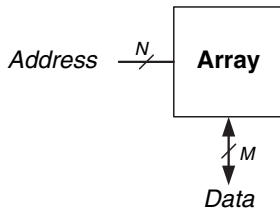


TECHNISCHE
UNIVERSITÄT
DARMSTADT

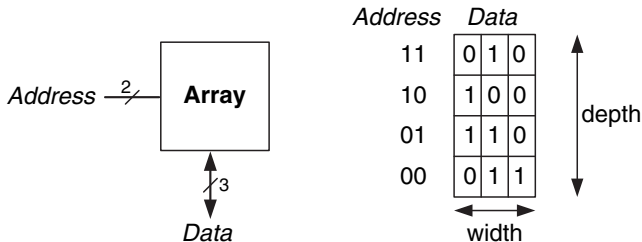
1. Einleitung
2. SystemVerilog für Zustandsautomaten
3. SystemVerilog Abschluss und Ausblick
4. Sequentielle Grundelemente
5. Speicherfelder
6. Zusammenfassung

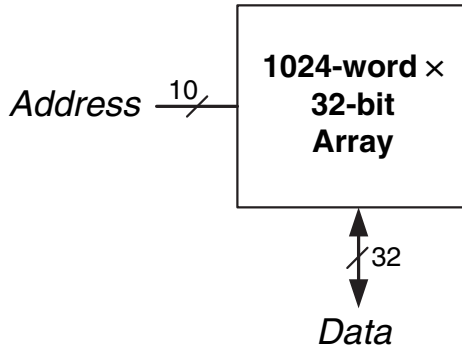
Anwendungs- software		Programme
Betriebs- systeme		Gerätetreiber
Architektur		Befehle Register
Mikro- architektur		Datenpfade Steuerung
Logik		Addierer Speicher
Digital- schaltungen		UND Gatter Inverter
Analog- schaltungen		Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen

- ▶ 2-dimensionales Array von Bitzellen
- ▶ Jede Bitzelle speichert ein Bit
- ▶ N Adressbits und M Datenbits:
 - ▶ 2^N Zeilen und M Spalten
 - ▶ **Tiefe:** Anzahl der Zeilen (Anzahl der Wörter)
 - ▶ **Breite:** Anzahl der Spalten (Wortbreite)
 - ▶ **Größe:** Tiefe \times Breite = $2^N \times M$ Bits



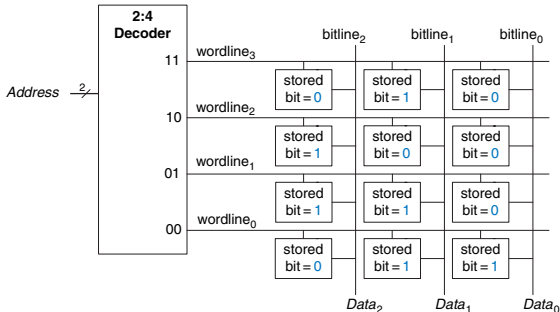
- ▶ $2^2 \times 3$ -Bit Array
- ▶ Anzahl der Wörter: 4
- ▶ Wortbreite: 3 Bits
- ▶ Beispiel: das 3-Bit Wort, das an der Adresse 10 gespeichert ist, lautet 100

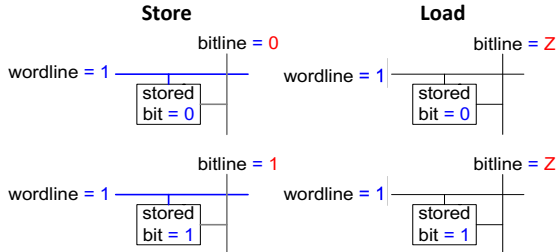
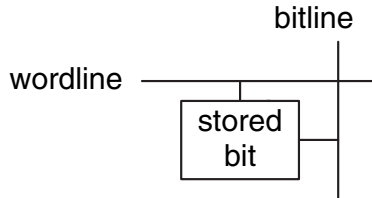




► Wordline:

- Vergleichbar zu *ENABLE* Signal
- Einzelne Zeile im Speicherfeld wird gelesen/geschrieben
- Entspricht einer eindeutigen Adresse
- Maximal eine Wordline kann HIGH sein














Agenda



TECHNISCHE
UNIVERSITÄT
DARMSTADT

1. Einleitung
2. SystemVerilog für Zustandsautomaten
3. SystemVerilog Abschluss und Ausblick
4. Sequentielle Grundelemente
5. Speicherfelder
6. Zusammenfassung

Anwendungs- software		Programme
Betriebs- systeme		Gerätetreiber
Architektur		Befehle Register
Mikro- architektur		Datenpfade Steuerung
Logik		Addierer Speicher
Digital- schaltungen		UND Gatter Inverter
Analog- schaltungen		Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen



- ▶ SystemVerilog für Zustandsautomaten
 - ▶ SystemVerilog Abschluss und Ausblick
 - ▶ Sequentielle Grundelemente
 - ▶ Speicherfelder
-
- ▶ Nächste Vorlesung behandelt
 - ▶ Speicherfelder (Fortsetzung), Logikfelder