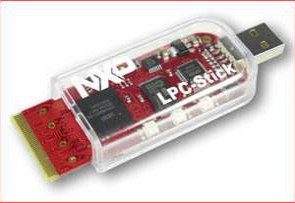


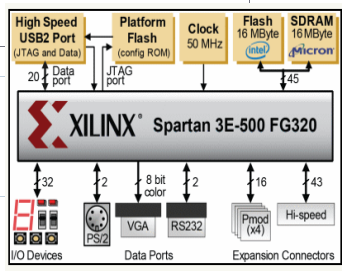
Informatik
HAW Hamburg



Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Computer Engineering WS 2012

Digitale Systeme
FSM



Prof. Dr. B. Schwarz

Informatik
HAW Hamburg

Digitale Systeme

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

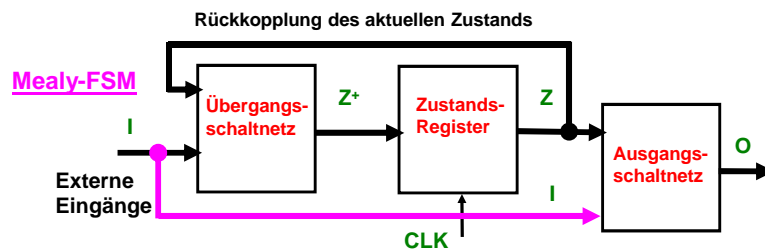
2. Entwurf und Verhalten synchroner Automaten
 - Automatenstrukturen
 - Entwurfsmethodik; Beispiel Sequenzerkennung
 - VHDL-Model, Timing, Synthese
 - Zwei-Prozess VHDL-Automatenbeschreibungen
 - Entkopplung von Zustandsautomaten
 - Entwurf eines sequentiellen Addierers (Mealy, Moore)



2. Entwurf und Verhalten synchroner Automaten

2.1 Automatenstrukturen

Mealy-Automat



Beim **Mealy-Automaten** gilt:

$$Z^+ = f(E, Z) \quad \text{und} \quad O = f_{\text{Mealy}}(I, Z)$$

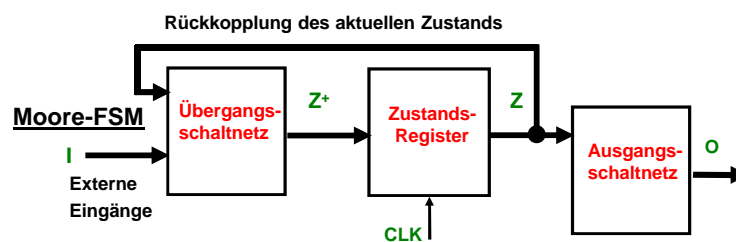
3

CE - DS 2 FSM

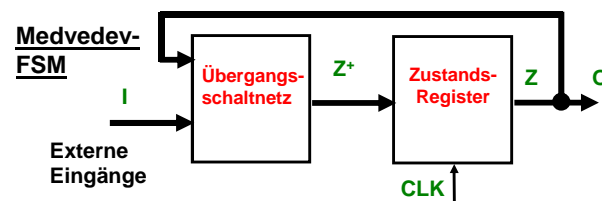


Moore- und Medvedev-Automaten

Beim Moore-Automaten gilt:
 $Z^+ = f(E, Z)$
 und
 $O = f_{\text{Moore}}(Z)$



Der Medvedev-Automat ist ein Moore-Automat, bei dem die Ausgänge den Zustands-Flipflops entsprechen:
 $Z^+ = f(E, Z)$
 und
 $O = Z$



4

CE - DS 2 FSM

2.2 Entwurfsmethodik ohne CAE

- 1) Erstelle ein Zustandsdiagramm. Ggf. kann zuerst auch eine mnemonische Folgezustands- und Ausgangstabelle erstellt werden.
- 2) Minimiere ggf. die Anzahl der Zustände
- 3) Wähle eine Menge von Zustandssignalen und ordne diesen die mnemonischen Zustände zu.
- 4) Wähle einen Flipflop-Typ für die Hardware-Realisierung (meistens D-FF).
- 5) Stelle Folgezustands- und Ausgangstabellen für die Zustands- bzw. Ausgangssignale auf.
- 6) Minimiere die Folgezustands- und Ausgangsfunktionen
- 7) Analysiere möglicherweise vorhandene Pseudozustände
- 8) Zeichne einen Schaltplan

5

CE - DS 2 FSM

Beispiel: Impulsfolgeerkennung

- In einem seriellen, 2 Bit breiten Datenstrom soll die Impulsfolge am Eingang **E = ...,01,11,10,...** erkannt werden und am Ausgang **A** des taktsynchronen Automaten mit einer '1' für die Dauer einer Taktperiode quittiert werden. Andernfalls soll der Ausgang '0' sein. Die Startheingangsimpulse können länger, als einen Takt anliegen.
- Nachfolgend werden die einzelnen Entwurfsschritte für eine Realisierung als Moore- und als Mealy-Automat erläutert.
- Für beide Varianten wird ein synthesefähiges VHDL-Modell erstellt.
- Das Zeitverhalten bei der Bildung des Folgezustands sowie des Ausgangssignals wird für beide Varianten analysiert.

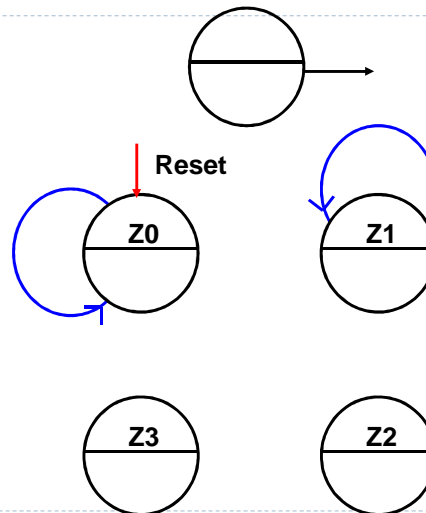
6

CE - DS 2 FSM

Moore-Automat

Zustände:

- Z0 : Anfangszustand ,
warte auf E="01".
Dies ist der Zustand
nach RESET.
- Z1: E="01" wurde
erkannt, warte auf
"11"
- Z2: E="11" wurde
erkannt, warte auf
"10"
- Z3: E="10" wurde
erkannt, gebe A='1'
aus.



7

CE - DS 2 FSM

Zustandskodierung, Folgezustands- u. Ausgangstabellen zur Sequenzerkennung

Heuristischer Ansatz für die
Zustandskodierung:

Zustand	Z ₁	Z ₀
Z0	0	0
Z1	0	1
Z2	1	0
Z3	1	1

Mit 2 DFFs existieren insgesamt
4! = 4*3*2*1 = 24 Permutationen

No.	Z3	Z2	Z1	Z0
1	00	01	10	11
2	00	01	11	10
3	00	10	01	11
4	00	10	11	01
...

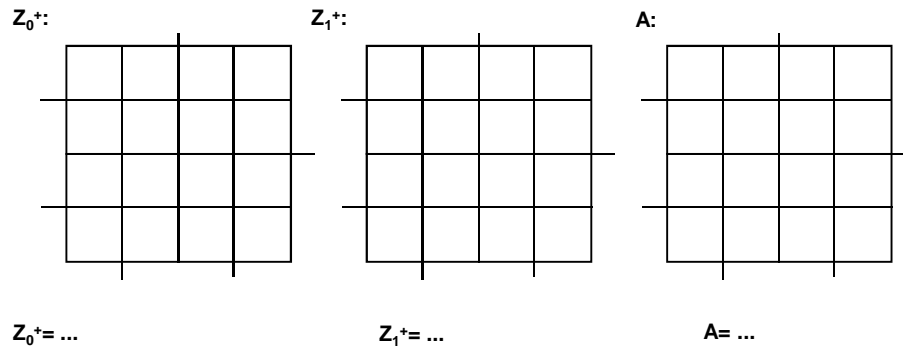
8

CE - DS 2 FSM

	Z ₁	Z ₀	E ₁	E ₀	Z ₁ ⁺	Z ₀ ⁺	A
Z0	0	0	0	0			
	0	0	0	1			
	0	0	1	0			
	0	0	1	1			
Z1	0	1	0	0			
	0	1	0	1			
	0	1	1	0			
	0	1	1	1			
Z2	1	0	0	0			
	1	0	0	1			
	1	0	1	0			
	1	0	1	1			
Z3	1	1	0	0			
	1	1	0	1			
	1	1	1	0			
	1	1	1	1			



KV-Minimierung der Übergangs- und Ausgangsschaltnetze



9

CE - DS 2 FSM




2.3 VHDL-Modell des Moore-Automaten

- Zusätzliche Anforderung: Das Fortschreiten des Automaten soll dann erfolgen, wenn das zusätzliche Freigabesignal **ENABLE = '1'** ist.
- Einfachster Ansatz: **Drei Funktionsblöcke** des Moore-Modells mit je einem Prozess beschrieben.


```
entity FSM_1_MOORE is
port( CLK, RESET, ENABLE : in bit;          -- sekundäre Eingangssignale
      E : in bit_vector(1 downto 0);        -- Impulsfolge
      A : out bit );                        -- Ausgangssignal
end FSM_1_MOORE;
architecture SEQUENZ of FSM_1_MOORE is
type ZUSTAENDE is (Z0, Z1, Z2, Z3);        -- Aufzählungstyp
signal ZUSTAND, FOLGE_Z: ZUSTAENDE;       -- Prozess-Kommunikation
begin
  Z_SPEICHER: process (CLK, RESET)         -- Zustandsaktualisierung
  begin
    if RESET = '1' then ZUSTAND <= Z0 after 5 ns;
    elsif CLK = '1' and CLK'event then
      if ENABLE = '1' then ZUSTAND <= FOLGE_Z after 5 ns;
      end if;
    end if;
  end process Z_SPEICHER;
```

10

CE - DS 2 FSM



Informatik
HAW Hamburg




Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

```


UE_SN: process (E, ZUSTAND)      -- Folgezustandsberechnung
begin
  case ZUSTAND is
    when Z0 => if E = "01" then FOLGE_Z <= Z1 after 5 ns;
               else FOLGE_Z <= Z0 after 5 ns;
               end if;
    when Z1 => if E = "11" then FOLGE_Z <= Z2 after 5 ns;
               elsif E = "01" then FOLGE_Z <= Z1 after 5 ns;
               else FOLGE_Z <= Z0 after 5 ns;
               end if;
    when Z2 => if E = "10" then FOLGE_Z <= Z3 after 5 ns;
               elsif E = "01" then FOLGE_Z <= Z1 after 5 ns;
               else FOLGE_Z <= Z0 after 5 ns;
               end if;
    when Z3 => if E = "01" then FOLGE_Z <= Z1 after 5 ns;
               else FOLGE_Z <= Z0 after 5 ns;
               end if;
  end case;
end process UE_SN;
A_SN: process (ZUSTAND)          -- Ausgangssignalberechnung
begin
  case ZUSTAND is
    when Z3 => A <= '1' after 5 ns;
    when others => A <= '0' after 5 ns;
  end case;
end process A_SN;
--end SEQUENZ;
  
```

CE - DS 2 FSM

11

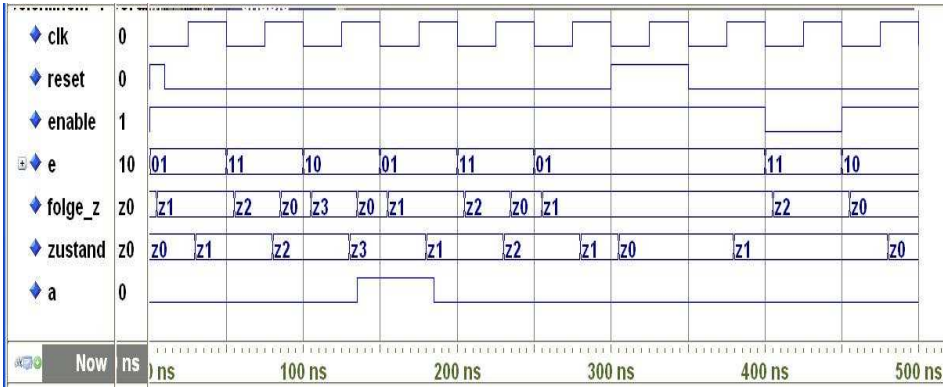


Informatik
HAW Hamburg



Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Zeitverhalten des Moore-Automaten



CE - DS 2 FSM

12

Moore-FSM: Implemented Equations

```

O <= (STATE(0).FBK.LFBK AND STATE(1).FBK.LFBK);
FDCPE_STATE0: FDCPE port map (STATE(0),STATE_D(0),CLK,RESET,'0');
STATE_D(0) <= ((NOT ENABLE AND STATE(0).LFBK)
OR (ENABLE AND NOT I(1) AND I(0))
OR (ENABLE AND I(1) AND NOT I(0) AND NOT STATE(0).LFBK
AND STATE(1).LFBK));

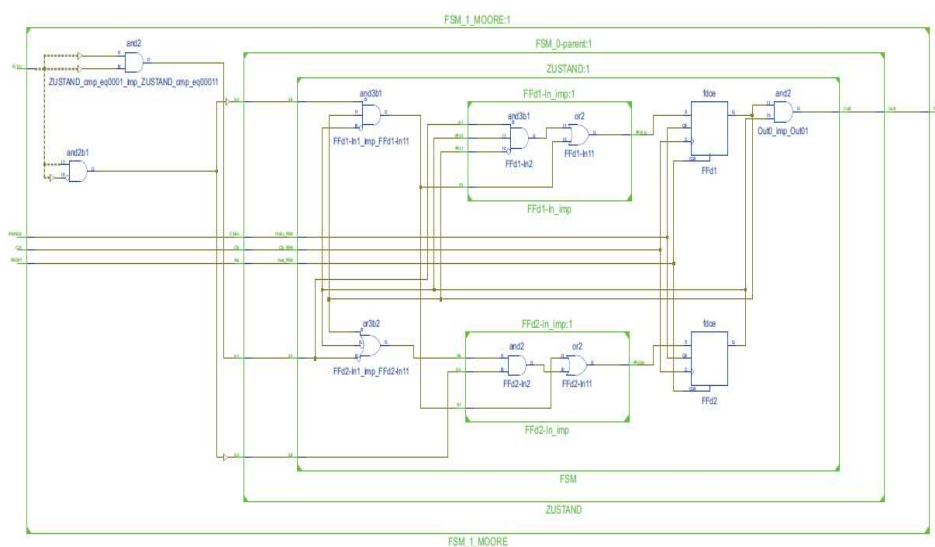
FDCPE_STATE1: FDCPE port map (STATE(1),STATE_D(1),CLK,RESET,'0');
STATE_D(1) <= ((NOT ENABLE AND STATE(1).LFBK)
OR (I(1) AND NOT I(0) AND NOT STATE(0).LFBK AND STATE(1).LFBK)
OR (ENABLE AND I(1) AND I(0) AND STATE(0).LFBK AND
NOT STATE(1).LFBK));
  
```

Register Legend: FDCPE (Q,D,C,CLR,PRE);

13

CE - DS 2 FSM

RTL Schematic





Synthese-Reportauszug

```

Synthesizing Unit <FSM_1_MOORE>.
Related source file is "C:\Users\Dr. B.
Schwarz/Documents/A_MSI_ISE_work/ce/fsm_1_moore/seq_moore.vhd".
Found finite state machine <FSM_0> for signal <ZUSTAND>.
-----
States          | 4
Transitions     | 10
Inputs          | 3
Outputs         | 1
Clock           | CLK                      (rising_edge)
Clock enable     | ENABLE                   (positive)
Reset           | RESET                   (positive)
Reset type       | asynchronous
Reset State     | z0
Power Up State  | z0
Encoding        | sequential
Implementation  | LUT
-----

Summary:
    inferred 1 Finite State Machine(s).
Unit <FSM_1_MOORE> synthesized.
*                               *
Advanced HDL Synthesis

=====
Optimizing FSM <ZUSTAND/FSM> on signal <ZUSTAND[1:2]> with sequential encoding.
-----
State | Encoding
-----
z0    | 00
z1    | 01
z2    | 10
z3    | 11

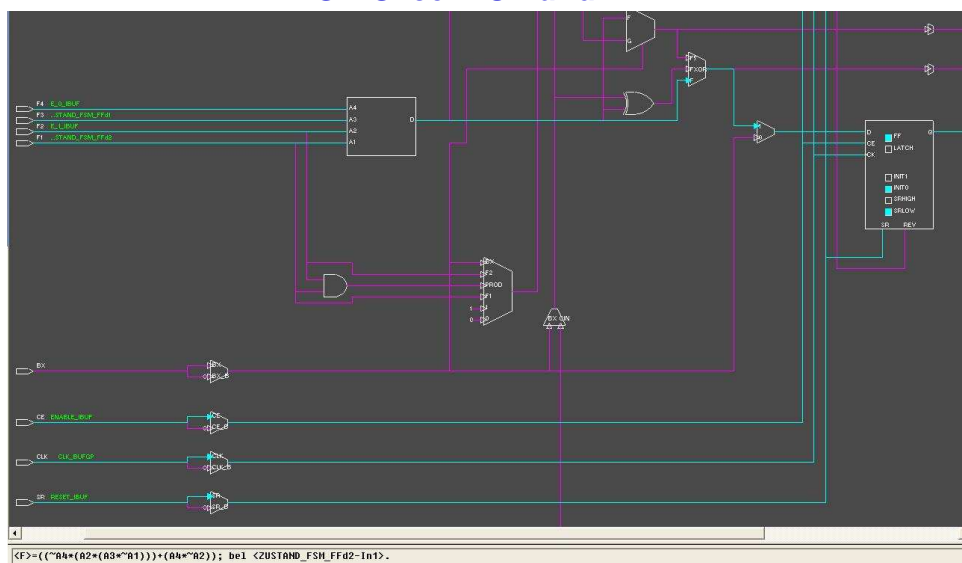
```

15

CE - DS 2 FSM



FPGA-Slice: LUT und DFF



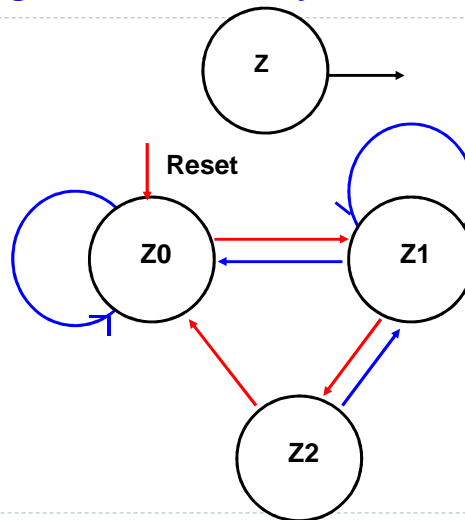
16

CE - DS 2 FSM



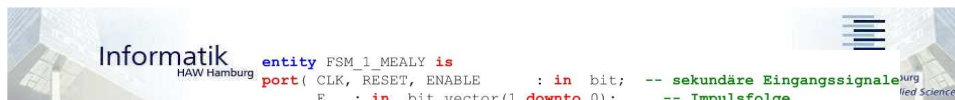
2.4 Impulsfolgeerkennung durch einen Mealy-Automaten

- Impulsfolge:
 $E = \dots 01, 11, 10, \dots$
- Da sich beim Mealy-Automaten das Eingangssignal E direkt auf das Ausgangssignal A auswirken kann, wird ein Zustand eingespart.
- Allerdings muss deshalb damit gerechnet werden, dass beim Ausgangssignal A des Mealy-Automaten Hazards auftreten!



17

CE - DS 2 FSM



VHDL-Modell der Mealy-FSM

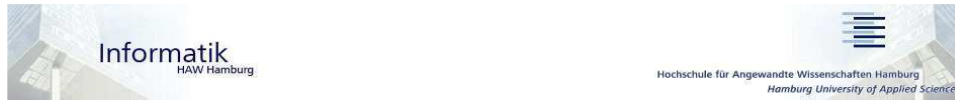
```

entity FSM_1_MEALY is
    port( CLK, RESET, ENABLE : in bit; -- sekundäre Eingangssignale
          E : in bit_vector(1 downto 0); -- Impulsfolge
          A : out bit ); -- Ausgangssignal
end FSM_1_MEALY;
architecture SEQUENZ of FSM_1_MEALY is
    type ZUSTAENDE is (Z0, Z1, Z2); -- Aufzählungstyp
    signal ZUSTAND, FOLGE_Z: ZUSTAENDE := Z2;
    attribute SAFE_RECOVERY_STATE: STRING;
    attribute SAFE_RECOVERY_STATE of ZUSTAND: signal is "Z1";
begin
    Z_SPEICHER: process(CLK, RESET) -- Zustandsaktualisierung
    begin
        if RESET = '1' then
            ZUSTAND <= Z1;
        else
            ZUSTAND <= FOLGE_Z;
        end if;
    end process;
    UE_SN: process(E, ZUSTAND) -- Folgezustandsberechnung
    begin
        FOLGE_Z <= Z0 after 5 ns; -- Defaultzuweisung
        case ZUSTAND is
            when Z0 => if E = "01" then FOLGE_Z <= Z1 after 5 ns;
                       else if E = "11" then FOLGE_Z <= Z2 after 5 ns;
                       elsif E = "01" then FOLGE_Z <= Z1 after 5 ns;
                       end if;
            when Z1 => if E = "11" then FOLGE_Z <= Z2 after 5 ns;
                       else if E = "01" then FOLGE_Z <= Z1 after 5 ns;
                       end if;
            when Z2 => if E = "01" then FOLGE_Z <= Z1 after 5 ns;
                       else if E = "11" then FOLGE_Z <= Z2 after 5 ns;
                       end if;
            when others => null;
        end case;
    end process;
    A_SN: process(E, ZUSTAND) -- Ausgangssignalsberechnung
    begin
        A <= '0' after 5 ns; -- Defaultzuweisung
        if (ZUSTAND = Z2 and E = "10") then A <= '1' after 5 ns;
        end if;
    end process;
end SEQUENZ;

```

18

CE - DS 2 FSM



Testbench instanziiert den Mealy-Automaten

Testobjekt als Instanz (DUT) in einer Entity ohne äußere Schnittstellen.

```
entity TEST_B_ME is -- Keine externen Signale
end TEST_B_ME;

architecture SEQUENZ of TEST_B_ME is
  component FSM_1_MEALY is
    port(CLK, RESET, ENABLE : in bit; -- sekundäre Eingangssignale
         E : in bit_vector(1 downto 0); -- Impulsfolge
         A : out bit); -- Ausgangssignal
  end component;

  signal CLK_I, RESET_I, ENABLE_I, A_I: bit; -- Interne Signale
  signal E_I: bit_vector(1 downto 0);
begin

  CLOCK: process -- Periodisches Taktsignal 20 MHz
  begin
    CLK_I <= '0'; wait for 25 ns;
    CLK_I <= '1'; wait for 25 ns;
  end process CLOCK;
```

19

CE - DS 2 FSM



Testbench instanziiert den Mealy-Automaten

Hazard in E = 10,00,10 bei:

```
ABLAUF: process -- keine Sensitivitätsliste; Stimuli-Abfolge
begin
  ENABLE_I <= '1'; RESET_I <= '1'; E_I <= "01"; wait for 50 ns;
  RESET_I <= '0'; wait for 100 ns;
  E_I <= "11"; wait for 30 ns;
  E_I <= "10"; wait for 20 ns; -- A_I <= '1'
  E_I <= "00"; wait for 15 ns; -- Hazard
  E_I <= "10"; wait for 85 ns; -- A_I <= '1'
  E_I <= "01"; wait for 50 ns;
  E_I <= "11"; wait for 50 ns; -- Z1 fest
  ENABLE_I <= '0'; wait for 50 ns;
  ENABLE_I <= '1'; wait for 50 ns;
  E_I <= "10"; wait for 50 ns;
  E_I <= "01"; wait for 50 ns;
end process ABLAUF;

DUT: entity work.FSM_1_MEALY(SEQUENZ) -- Instanziierung der Mealy FSM
  port map(CLK => CLK_I, RESET => RESET_I, ENABLE => ENABLE_I,
           E => E_I, A => A_I); -- formal => actual
end SEQUENZ;
```

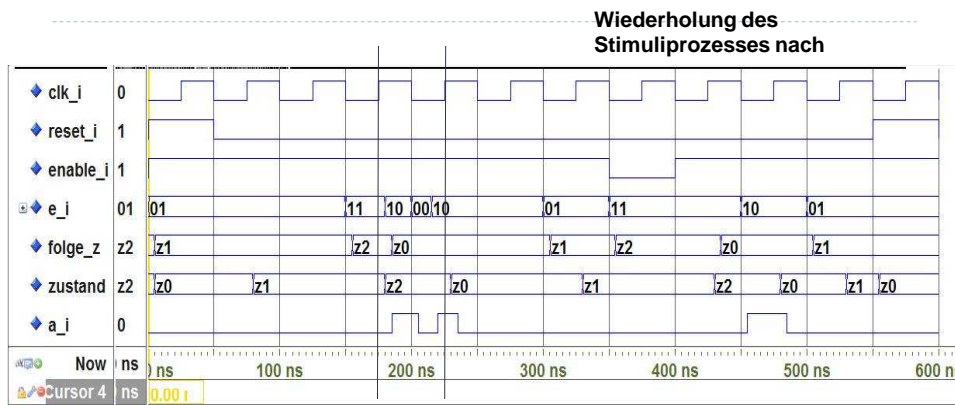
Wiederholung der Stimulisequenz ab:

20

CE - DS 2 FSM



VHDL-Simulation der Mealy-FSM



Hazard in E = 10,00,10

21

CE - DS 2 FSM



Synthese-Reportauszug

```

Synthesizing Unit <FSM_1_MEALY>.
Related source file is "C:/Users/.../Documents/A_MSI_ISE_work/ModelSim_source/Lectures/cel/seq_mealy.vhd".
Found finite state machine <FSM_0> for signal <ZUSTAND>.
-----
| States           | 3 |
| Transitions     | 7 |
| Inputs          | 2 |
| Outputs         | 1 |
| Clock           | CLK           | (rising_edge)
| Clock enable    | ENABLE        | (positive)
| Reset           | RESET         | (positive)
| Reset type      | asynchronous
| Reset State     | z0
| Power Up State  | z2
| Recovery State  | z1
| Encoding        | sequential
| Implementation  | LUT
-----

Summary:          inferred  1 Finite State Machine(s).
Unit <FSM_1_MEALY> synthesized.
=====
*               Advanced HDL Synthesis               *
=====
Optimizing FSM <ZUSTAND/FSM> on signal <ZUSTAND[1:2]> with sequential encoding.

State | Encoding
-----
z0    | 10
z1    | 01
z2    | 00

```

22

CE - DS 2 FSM

Mealy-FSM: Implemented Equations

$A \leq (E(1) \text{ AND NOT } E(0) \text{ AND NOT ZUSTAND_FSM_FFd2.LFBK AND NOT ZUSTAND_FSM_FFd1.LFBK});$

FDCPE_ZUSTAND_FSM_FFd1: FDCPE

port map (ZUSTAND_FSM_FFd1,ZUSTAND_FSM_FFd1_D,CLK,'0',RESET);

ZUSTAND_FSM_FFd1_D <= ((NOT E(0) AND ENABLE) OR
(NOT ENABLE AND ZUSTAND_FSM_FFd1.LFBK) OR
(E(1) AND ENABLE AND NOT ZUSTAND_FSM_FFd2.LFBK));

FDCPE_ZUSTAND_FSM_FFd2: FDCPE

port map (ZUSTAND_FSM_FFd2,ZUSTAND_FSM_FFd2_D,CLK,RESET,'0');

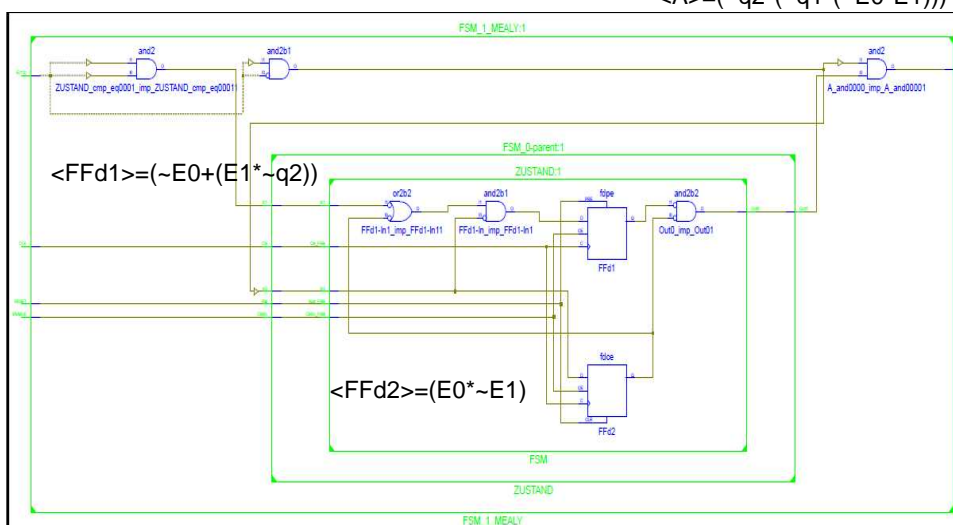
ZUSTAND_FSM_FFd2_D <= ((NOT ENABLE AND ZUSTAND_FSM_FFd2.LFBK) OR
(NOT E(1) AND E(0) AND ENABLE));

23

CE - DS 2 FSM

Mealy-FSM RTL Schematic

$\langle A \rangle = (\sim q2 * (\sim q1 * (\sim E0 * E1)))$



24

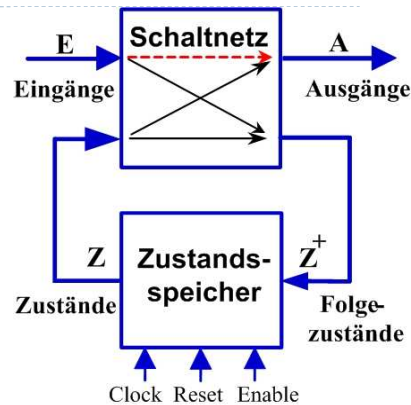
CE - DS 2 FSM



2.5 Zwei-Prozess VHDL Automatenbeschreibung

Ausgangspunkt vereinfachter Automatenmodelle ist die Huffman-Normalform.

- Ein Zustandsspeicher
- Ein kombiniertes Schaltnetz, das das Übergangs- und das Ausgangsschaltnetz so zusammenfasst, dass die Zustandsabfrage nur einmal realisiert wird.
- Eine direkte Verbindung der Eingänge **E** auf die Ausgänge **A** (**gestrichelt**) existiert nur, falls ein **Mealy-Verhalten** modelliert werden soll.
- Entsprechend lassen sich VHDL-Modelle von Zustandsautomaten auch mit zwei Prozessen realisieren. Medvedev-Automaten können sogar mit einem einzigen Prozess aufgebaut werden.



25

CE - DS 2 FSM



Kombiniertes ÜSN und ASN

```

UE_A_SN: process (E, ZUSTAND) -- Folgezustands- u. Ausgangsberechnung
begin
    A <= '0' after 5 ns;
    FOLGE_Z <= Z0 after 5 ns; -- Defaultzuweisungen
    case ZUSTAND is
        when Z0 => if E = "01" then
                     FOLGE_Z <= Z1 after 5 ns;
                   end if;
        when Z1 => if E = "11" then
                     FOLGE_Z <= Z2 after 5 ns;
                   elsif E = "01" then
                     FOLGE_Z <= Z1 after 5 ns;
                   end if;
        when Z2 => if E = "10" then
                     FOLGE_Z <= Z3 after 5 ns;
                   elsif E = "01" then
                     FOLGE_Z <= Z1 after 5 ns;
                   end if;
        when Z3 => A <= '1' after 5 ns; -- Moore-Ausgang
                   if E = "01" then
                     FOLGE_Z <= Z1 after 5 ns;
                   end if;
    end case;
end process UE_A_SN;
--end SEQUENZ;

```

26

CE - DS 2 FSM



2.6 Entkopplung von Zustandsautomaten

- Die maximale Taktfrequenz eines synchronen digitalen Systems wird durch die längste Laufzeit eines Signals durch die kombinatorische Logik zwischen je zwei Flipflops bestimmt (kritischer Pfad): RTL



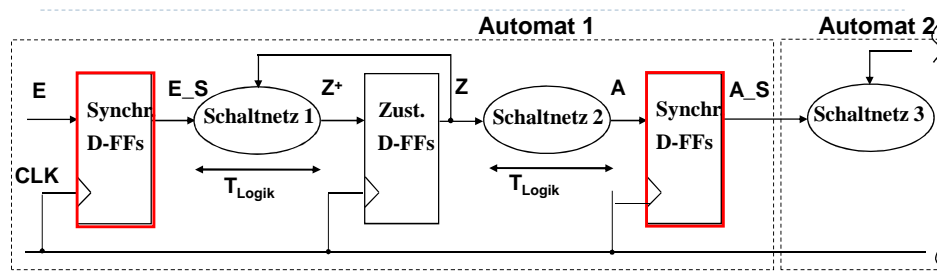
- Bei gekoppelten Automaten ergibt sich die Laufzeit durch den kombinatorischen Pfad als Summe der Laufzeiten durch das Ausgangsschaltnetz des ersten Automaten und der durch das Übergangsschaltnetz des zweiten Automaten!

27

CE - DS 2 FSM



Ein- und Ausgangssignalsynchronisation



Maximale Taktfrequenz:

T_{PD} : D-Flipflop Verzögerung (CLK → Ausgang Q)

T_{Logik} : Signallaufzeit auf dem längsten kombinatorischen Pfad incl. Verdrahtungspfade.

T_S : Einzuhaltende Setup-Zeit der Flipflop-Dateneingänge

28

CE - DS 2 FSM

Moore-FSM mit synchronisierten Schnittstellen

```
entity FSM_sync is
    port(
        CLK, RESET : in bit;
        E: in bit_vector(1 downto 0);
        A_S: out bit );
end FSM_sync;

architecture SEQUENZ of FSM_sync is
    type ZUSTAENDE is (Z0, Z1, Z2, Z3);
    signal ZUSTAND, FOLGE_Z: ZUSTAENDE;
    signal E_S: bit_vector(1 downto 0);
    signal A: bit;
begin
    SYNC: process (CLK, RESET)
    begin
        if RESET = '1' then
            E_S <= (others=>'0') after 5 ns;
            A_S <= '0' after 5 ns;
        elsif CLK='1' and CLK'event then
            E_S <= E after 5 ns;
            A_S <= A after 5 ns;
        end if;
    end process SYNC;
```

-- Synchr. Ausgangssignal

-- Synchr. Eingangssignal

-- Async. Ausgangssignal

-- E/A-Synchronisation

29

CE - DS 2 FSM

Huffman-Normalform mit synchronisierten Eingängen

```
Z_SPEICHER: process (CLK, RESET) -- Zustandsaktualisierung
begin
    if RESET = '1' then
        ZUSTAND <= Z0 after 5 ns;
    elsif CLK = '1' and CLK'event then
        ZUSTAND <= FOLGE_Z after 5 ns;
    end if;
end process Z_SPEICHER;

UE_A_SN: process (E_S, ZUSTAND) -- Folgezustands- u. Ausgangsberechnung
begin
    ...
    ...
end process UE_A_SN;
end SEQUENZ;
```

-- vgl. Huffman Moore Modell, allerdings

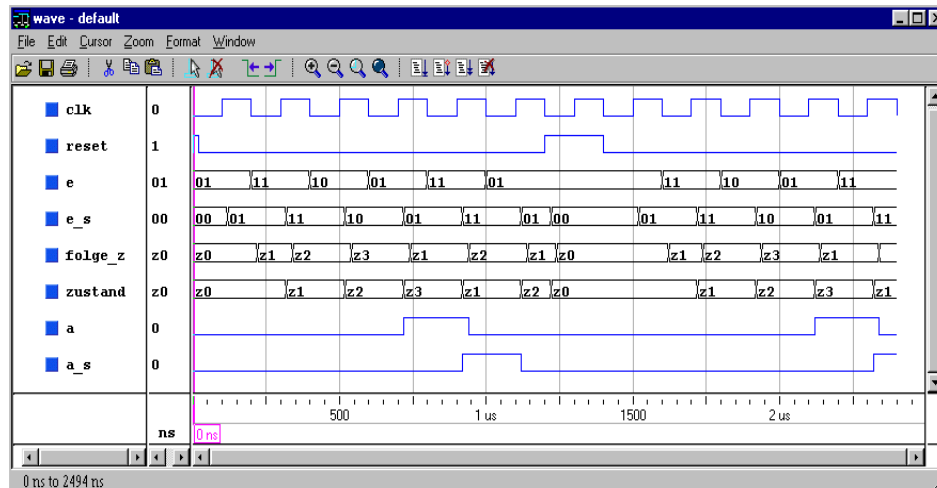
-- muss E_S abgefragt werden

30

CE - DS 2 FSM



Zeitverhalten des Moore-Automaten mit Eingangs- und Ausgangssynchronisation



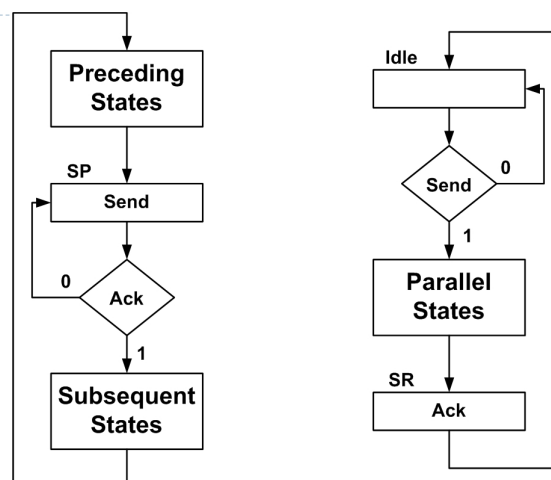
31

CE - DS 2 FSM



Kommunizierende parallele Systeme

- System-Partitionierung in kleinere Subsysteme mit reduzierter Teilkomplexität.
- Sub-FSMs werden von Main-FSM an mehreren Stellen aufgerufen.
- Z. B.: Client-Server Aufgabenverteilung.
- Die Signale Send und Acknowledge werden nur gesetzt, wenn der jeweilige Zustand SP bzw. SR aktiv ist.
- Direkte Kopplung von synchronen Moore-FSM mit gleichem Taktsignal ohne zusätzliche Maßnahmen realisierbar.



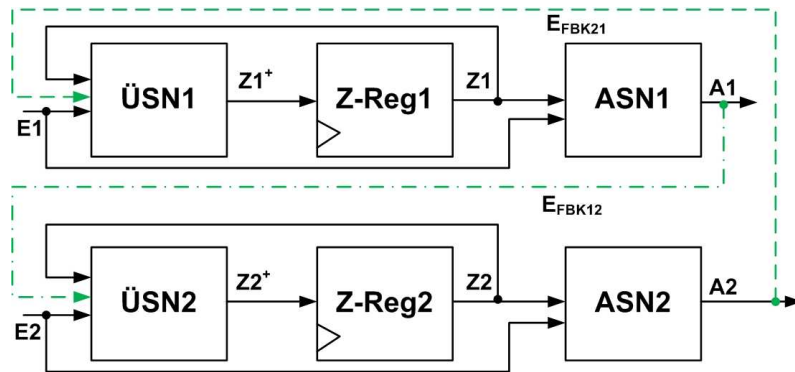
32

CE - DS 2 FSM



Gekoppelte Mealy-FSM

- Wenn bei der Kopplung von Mealy-Automaten ein Ausgang des 2. Automaten auf den Eingang des 1. Automaten direkt zurück gekoppelt wird, so entsteht eine kombinatorische Schleife → **die Ausgänge können schwingen!**



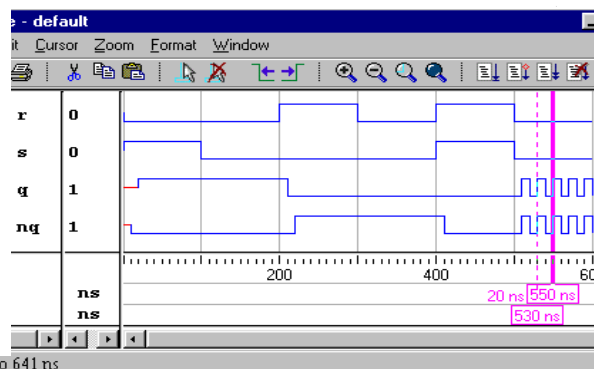
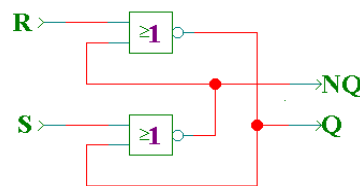
33

CE - DS 2 FSM



Einfache kombinatorische Schleife

- Im Übergang vom irregulären in den Halten-Zustand treten bei einem **NOR-RS-Latch** Schwingungen auf, die eine Periode mit $2 \cdot \text{Gatterlaufzeit}$ haben.
- Irregulär Reset=Setzen=1:
 $Q=NQ=0$,
 $R=S=1$ dominiert
- Halten $R=S=0$ u. $Q=NQ=0$:
 $\Rightarrow Q=NQ=1$
 \Rightarrow Rückführung dominiert
 $\Rightarrow Q=NQ=0$

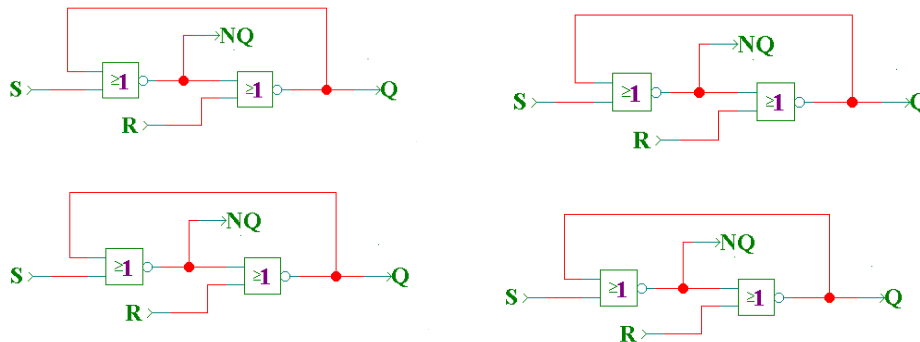


34



Analyse der Zustandsübergänge im Basis RS-Latch

NOR-Gatter: Ein beliebiger '1' Eingang erzwingt eine '0' am Ausgang
Alle Eingänge '0' bewirken eine '1' am Ausgang



35

CE - DS 2 FSM



2.7 Sequentieller Addierer

- Die erste Hauptaufgabe eines Automatenentwurfs liegt bei der Umsetzung einer **textuellen Spezifikation in ein Zustandsdiagramm**.
- Dazu ist zunächst zu prüfen:
 - Welche Eingangssignale sind synchron, welche asynchron?
 - Wie viele Zustände sind erforderlich, und welche Bedeutung haben diese?
 - Muss der Automat (aus Geschwindigkeitsgründen) als Mealy-Automat realisiert werden oder reicht ein Moore-Automat mit einem Takt mehr Latenz?
 - Ist es erforderlich, die Anzahl der Zustände in einem zweiten Schritt systematisch zu minimieren?
 - Welche Zielhardware (FPGA oder (C)PLD) ist vorgesehen?
 - Ist für die Anwendung eine sichere Rückkehr aus möglicherweise vorhandenen Pseudozuständen sicher zustellen?
- Bei der Erstellung des Zustandsdiagramms werden zunächst die "normalen" Zustandsübergänge betrachtet und hinterher die Sonderfälle.
- Der sequentielle Addierer ist ein Beispiel für eine in den Automaten integrierte Arithmetikfunktion: schnelle **kombinierte Lösung** für kleine Aufgaben.

36

CE - DS 2 FSM



Einzelschrittaddition: Parallel-Seriell-Umsetzer + FSM

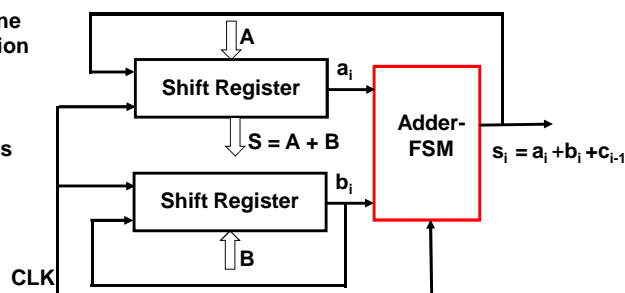
- Der Addierer besteht aus zwei Schieberegistern, in denen die Operandenbits a_i und b_i takt synchron nach rechts geschoben werden. In der FSM erfolgt eine 1-Bit Addition der jeweils beiden niederwertigen Operanden a_i , b_i und des Carry-Bits c_{i-1} der vorangegangenen Addition. Das Summationsbit s_i wird im Ergebnis-Schieberegister von links nach rechts geschoben.

Vorteile:

Mit einem 1-Bit Volladdierer kann eine platzsparende Addition auch für große Bitbreiten erfolgen. Die Addierer-FSM speichert implizit das Carry-Bit der 1-Bit-Additionen.

Nachteil:

Eine n-Bit Addition erfordert n-Takte.



37

CE - DS 2 FSM



Entwurf eines Mealy-Automaten

Abhängig vom Wert des Carry-Bits c_{i-1} der jeweils vorherigen 1-Bit Addition realisiert die FSM unterschiedliche Ergebnisse und Zustandsübergänge.

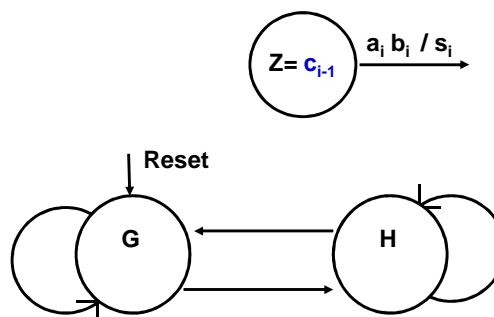
Bedeutung der Zustände:

G: Carry-In = '0'

H: Carry-In = '1',

Zustandsfolge- und Ausgangstabelle:

Z=	a_i	b_i	Z+=	s_i
G	0	0		
G	0	1		
G	1	0		
G	1	1		
H	0	0		
H	0	1		
H	1	0		
H	1	1		



38

CE - DS 2 FSM

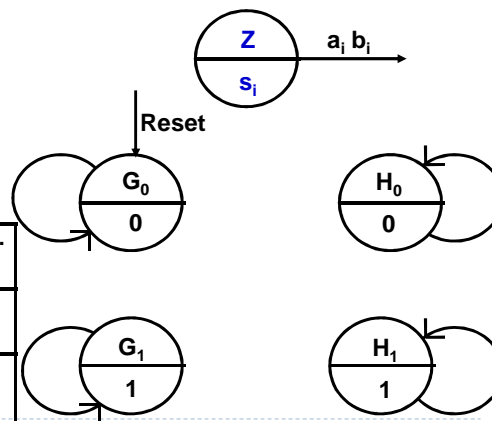
Entwurf eines Moore-Automaten

Beim Moore-Automat darf das **Ausgangssignal** s_i nur vom Zustand $Z = c_{i-1}$ abhängen → Aufspaltung der Mealy-Zustände G und H in je zwei Zustände G_0, G_1 bzw. H_0, H_1 .

Gewählte Zustandskodierung:

Z	Q1	Q0
G_0	0	0
G_1	0	1
H_0	1	0
H_1	1	1

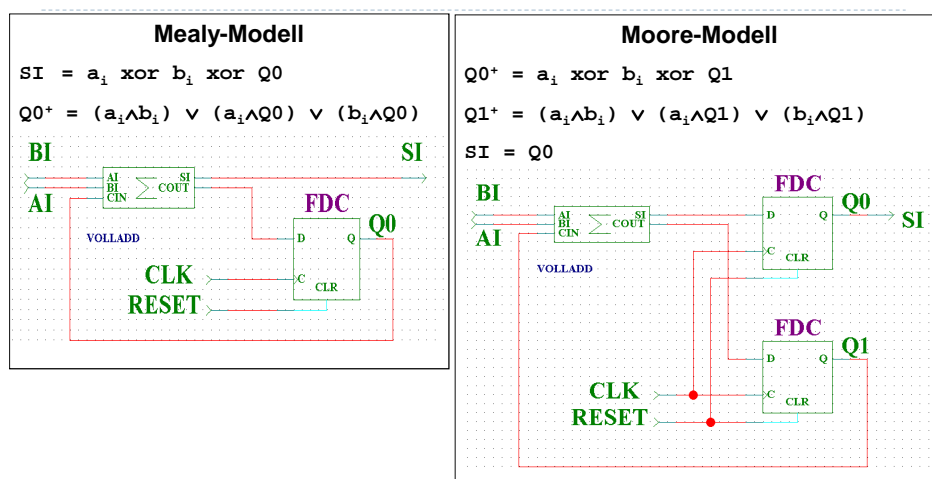
Zustand Z	Folgezustand Z^+				Ausg. s_i
	$a_i b_i = 00$	01	10	11	
G_0					
G_1					
H_0					
H_1					



39

CE - DS 2 FSM

Syntheseergebnis der sequentiellen Addierer



40

CE - DS 2 FSM

