Universität Paderborn
Institut für *Elektrotechnik und Informationstechnik*
Fachgebiet *Datentechnik*
Prof. Sybille Hellebrand

Klausur

Grundlagen der technischen Informatik

11. August 2017

Punkteverteilung						
Aufgabe	1	2	3	4	5	Σ
maximale Punkte	10	20	20	20	20	90
erreichte Punkte						

Note:	

Aufkleber

Name:	
Matrikelnummer:	
Studienrichtung:	

Studium Generale: ☐ Ja
☐ Nein

Hinweise:

Für die Lösung der Klausuraufgaben sind ausschließlich die Aufgabenblätter zu verwenden. Lösungsangaben außerhalb der Aufgabenblätter („Schmierzettel“, etc.) werden bei der Bewertung nicht berücksichtigt!

Beschriften Sie jede Doppelseite mit Ihrer Matrikelnummer!

Mit Bleistift oder der Korrekturfarbe rot angefertigte Lösungen werden nicht bewertet!

Die Verwendung von „Tipp-Ex“ oder „Tintenkiller“ ist untersagt.

Es ist ein handgeschriebener DIN-A4 Zettel als Hilfsmittel zugelassen!

Es sind keine weiteren Hilfsmittel zugelassen!

Die Aufgabe 5 muss von Studierenden, welche die Veranstaltung als Studium Generale gewählt haben nicht bearbeitet werden!

Aufgabe 1: (Darstellung von Informationen) 10 Punkte

a) Stellen Sie die Dezimalzahl 42_{10} als Binärzahl ohne Vorzeichen dar.
Kreuzen Sie die richtige Lösung an. (2 Punkte)

- ☐ 101010_2
- ☐ 010101_2
- ☐ 1000010_2
- ☐ Keine Lösung ist richtig.

b) Gegeben sei die Binärzahl 11001101_{2C} in 2er-Komplement Darstellung. Welche Dezimalzahl repräsentiert diese Binärzahl?
Kreuzen sie die richtige Lösung an. (2 Punkte)

- ☐ -205_{10}
- ☐ 205_{10}
- ☐ -51_{10}
- ☐ 51_{10}
- ☐ Keine Lösung ist richtig.

c) Gegeben sei die Hexadezimalzahl $C3F0_{16}$. Geben Sie die Zahl in Binärdarstellung an.
Kreuzen Sie die richtige Lösung an. (2 Punkte)

- ☐ $0011\ 1100\ 1111\ 0000_2$
- ☐ $1100\ 0011\ 1111\ 0000_2$
- ☐ $1011\ 0011\ 1111\ 0000_2$
- ☐ Keine Lösung ist richtig.

d) Gegeben sei die Dezimalzahl $-3{,}5_{10}$. Geben Sie die Zahl in IEEE 754 Gleitkommaformat an.

Kreuzen Sie die richtige Lösung an. (2 Punkte)

- ☐ $1\ 10000000\ 11100000000000000000000_2$
- ☐ $1\ 00000001\ 11000000000000000000000_2$
- ☐ $1\ 00000001\ 11100000000000000000000_2$
- ☐ $1\ 10000000\ 11000000000000000000000_2$
- ☐ Keine Lösung ist richtig.

e) Die C Standardbibliothek `stdint.h` stellt den Datentyp `int32_t` zur Verfügung. Für die Darstellung einer Zahl in diesem Datentyp wird die 2er-Komplement Darstellung einer 32-Bit Binärzahl verwendet. Welcher Wertebereich kann mit diesem Datentypen abgebildet werden?

Kreuzen Sie die richtige Lösung an. (2 Punkte)

- ☐ $[-(2^{31}-1), 2^{31}-1]$
- ☐ $[-2^{31}, 2^{31}-1]$
- ☐ $[-2^{32}, 2^{32}-1]$
- ☐ $[-(2^{32}-1), 2^{32}-1]$
- ☐ Keine Lösung ist richtig.

Aufgabe 2: (Logikoptimierung) 20 Punkte

Gegeben sei die Logikschaltung aus Abbildung 1.

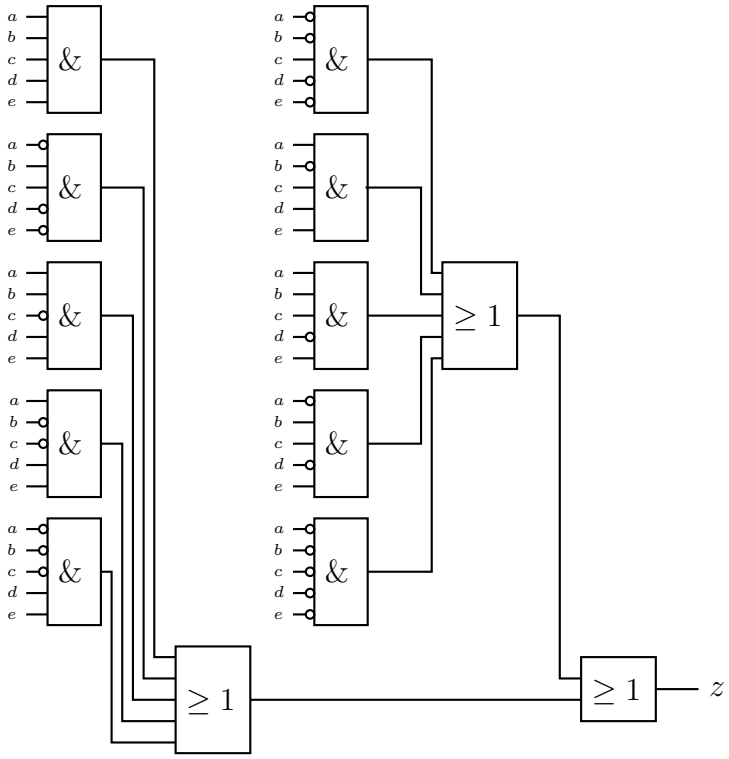


Abbildung 1: Eine Logikschaltung

a) Bestimmen Sie die Einsstellenmenge der Schaltungsfunktion $z = f(a,b,c,d,e)$: (2 Punkte)

$\mathcal{E} = \{$ _____

_____ $\}$

b) Gegeben sei die Einsstellenmenge

$$\mathcal{E} = \{11000,11100,01011,11011,00011,\\ 00111,10011,00001,01001,00000\}$$

Bestimmen Sie die Primimplikanten $\mathcal{P}$ von $\mathcal{E}$ mit dem Quine-McCluskey Verfahren. Füllen Sie dazu die Tabelle aus. (13 Punkte)

L_0	✓						

$\mathcal{P}$={_____}

Ersatztabelle. Ungültige Lösungen streichen!

L_0	✓						

c) Gegeben sei die Logikfunktion $z = abc + abe + bce + cde$. Eine Implementierung der Schaltung mit 3-AND und 2-OR Gattern ist in Abbildung 2 dargestellt.

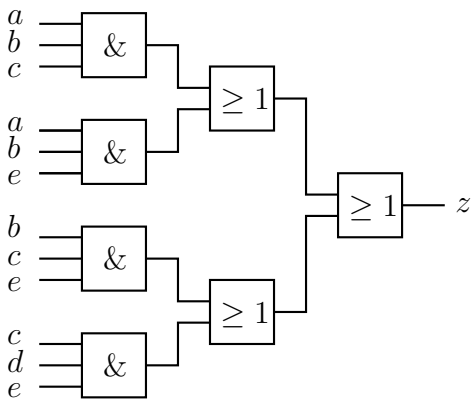


Abbildung 2: Implementierung von z.

Zeichnen Sie eine äquivalente Schaltung, welche ausschließlich 3-NAND und 2-NAND Gatter benötigt.

Hinweis: Die volle Punktzahl wird für eine Variante erreicht, welche höchstens genau so viele Gatter benötigt wie die Implementierung aus Abbildung 2. Für Varianten mit mehr Gattern können 3 Punkte erreicht werden. (5 Punkte)

Aufgabe 3: (Automaten) 20 Punkte

Gegeben ist ein durch folgende Automatentabelle (Tabelle 1) spezifizierter Mealy-Automat:

δ/λ	a	b	c
A	$B/0$	$C/1$	$E/1$
B	$F/1$	$E/1$	$B/0$
C	$B/0$	$A/1$	$F/1$
D	$D/1$	$E/1$	$B/0$
E	$D/1$	$D/1$	$C/0$
F	$D/1$	$D/1$	$A/0$

Tabelle 1: Automatentabelle eines Mealy-Automaten

a) Bestimmen Sie den äquivalenten zustandsminimalen Mealy-Automaten mit Hilfe des Ginsburg/Huffman-Verfahrens. (10 Punkte)

δ/λ	a	b	c

δ/λ	a	b	c

δ/λ	a	b	c

δ/λ	a	b	c

δ/λ	a	b	c

δ/λ	a	b	c

Ersatztabellen. Ungültige Lösung streichen!

b) Geben Sie für den folgenden Automaten (Tabelle 2) den Mealy-Automatengraphen an. (5 Punkte)

δ/λ	1	0
A	$B/1$	$D/0$
B	$A/0$	$C/1$
C	$D/1$	$B/1$
D	$B/0$	$D/1$

Tabelle 2: Automatentabelle

Graph des Mealy-Automaten

Ersatzgraph. Ungültige Lösung streichen!

c) Geben Sie für den Automaten aus Aufgabenteil **b)** (Tabelle 2) den äquivalenten Moore-Automatengraphen an. (5 Punkte)

Graph des Moore-Automaten

Ersatzgraph. Ungültige Lösung streichen!

Aufgabe 4: (RTL-Entwurf) 20 Punkte

a) Berechnen Sie per Division mit Rückspeichern den Quotienten a/b für die folgenden fünf-bit Zahlen: $a = 10101_2, b = 00101_2$. Füllen Sie hierzu das Schema in Abbildung 3 aus, in welchem A und B bereits die entsprechenden Operanden enthalten.(10 Punkte)

B					
0	0	0	1	0	1

P						A					Nächste Aktion
0	0	0	0	0	0	1	0	1	0	1	

Abbildung 3: Schema zur Division mit Rückspeichern

B					
0	0	0	1	0	1

P						A					Nächste Aktion
0	0	0	0	0	0	1	0	1	0	1	

Abbildung 4: **Ersatzschema, ungültige Lösung streichen!**

b) Abbildung 5 zeigt den Datenpfad und die Steuerung eines sequentiellen Dividierers (mit Rückspeichern) für n-bit Zahlen. Über den INBUS sollen nacheinander die beiden Zahlen a und b geschickt und in Register A und B geschrieben werden. Das ganzzahlige Ergebnis der Division a/b soll anschließend in Register A stehen, während Register P den Rest der Division enthält. Beide Registerinhalte (A und P) sollen nach der Berechnung nacheinander auf den OUTBUS gelegt werden. Tragen Sie alle notwendigen Kontrollpunkte in Abbildung 5 ein und beschreiben Sie die Funktion jedes Kontrollpunktes kurz in der nachfolgenden Tabelle. (5 Punkte)

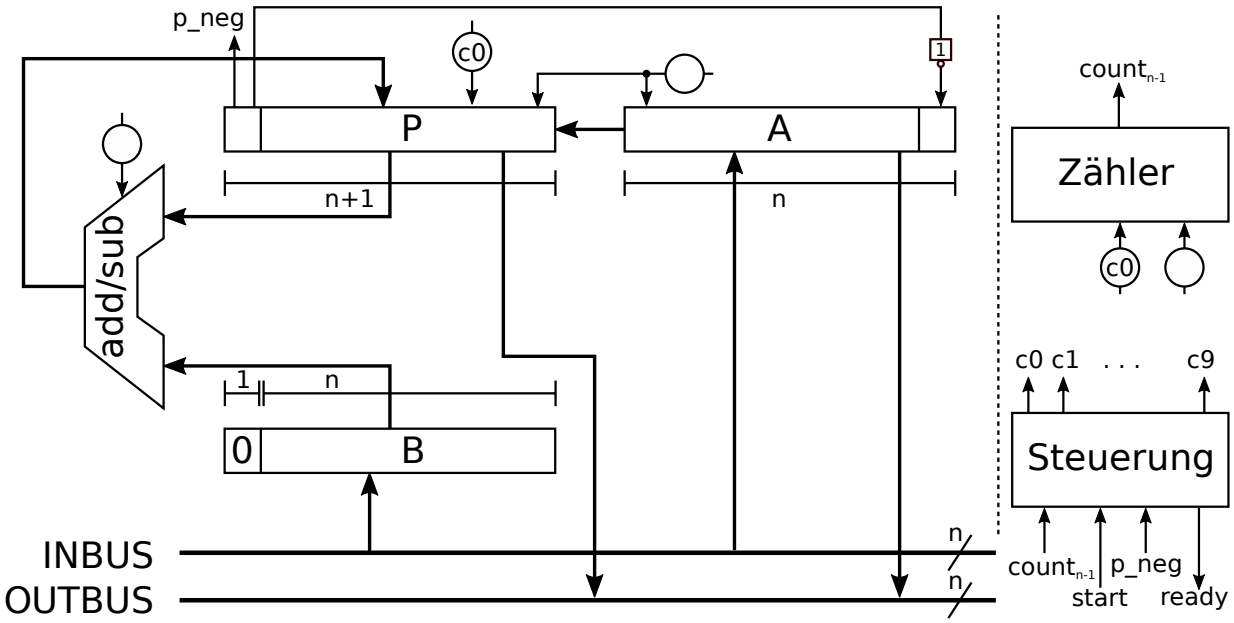


Abbildung 5: Unvollständiger Datenpfad eines Dividierers mit Steuerung

Kontrollpunkte:

c0: Register P und Zähler wird mit 0 initialisiert

c1: _____

c2: _____

c3: _____

c4: _____

c5: _____

c6: _____

c7: _____

c8: _____

c9: _____

c) Abbildung 6 zeigt nun die Steuerung für den sequentiellen Dividierer aus Aufgabenteil **b)** in Form eines Moore-Automaten. Ergänzen Sie den Automaten um die fehlenden Zustandsübergangsbedingungen und den zugehörigen Steuersignalen aus Aufgabenteil **b)**. (5 Punkte)

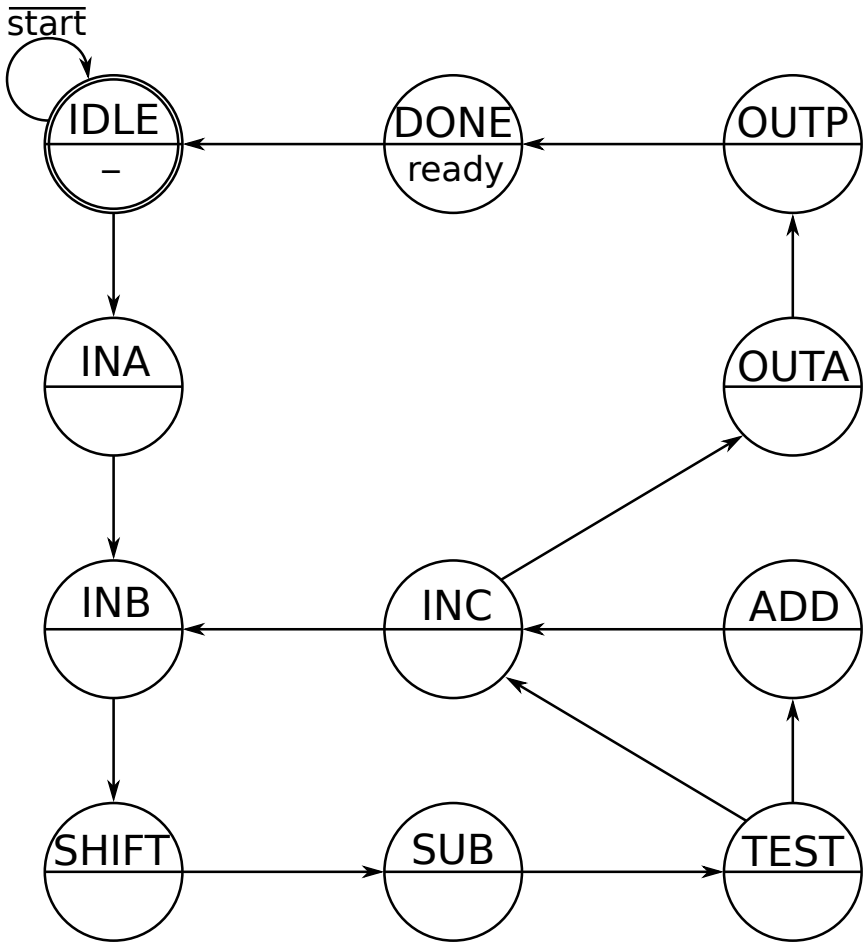


Abbildung 6: Automat einer Dividierer-Steuerung (Moore)

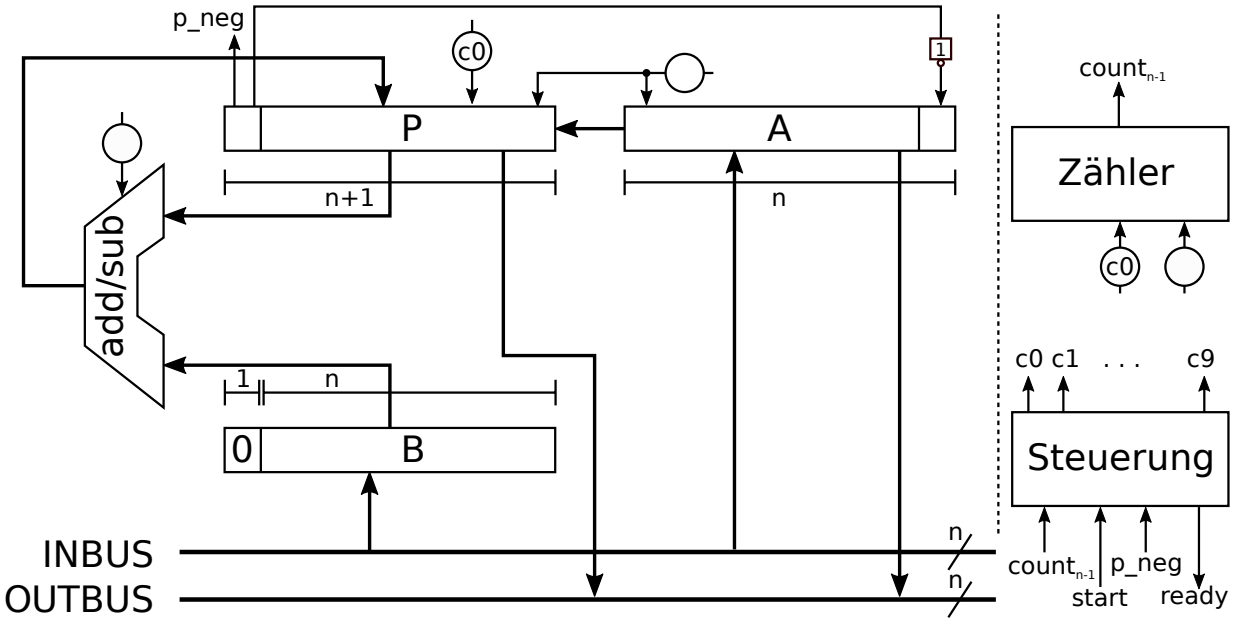


Abbildung 7: **Ersatz Dividierer-Datenpfad, ungültige Lösung streichen!**

Ersatztabelle für Kontrollpunkte, ungültige Lösung streichen!

c0: Register P und Zähler wird mit 0 initialisiert

c1: _____

c2: _____

c3: _____

c4: _____

c5: _____

c6: _____

c7: _____

c8: _____

c9: _____

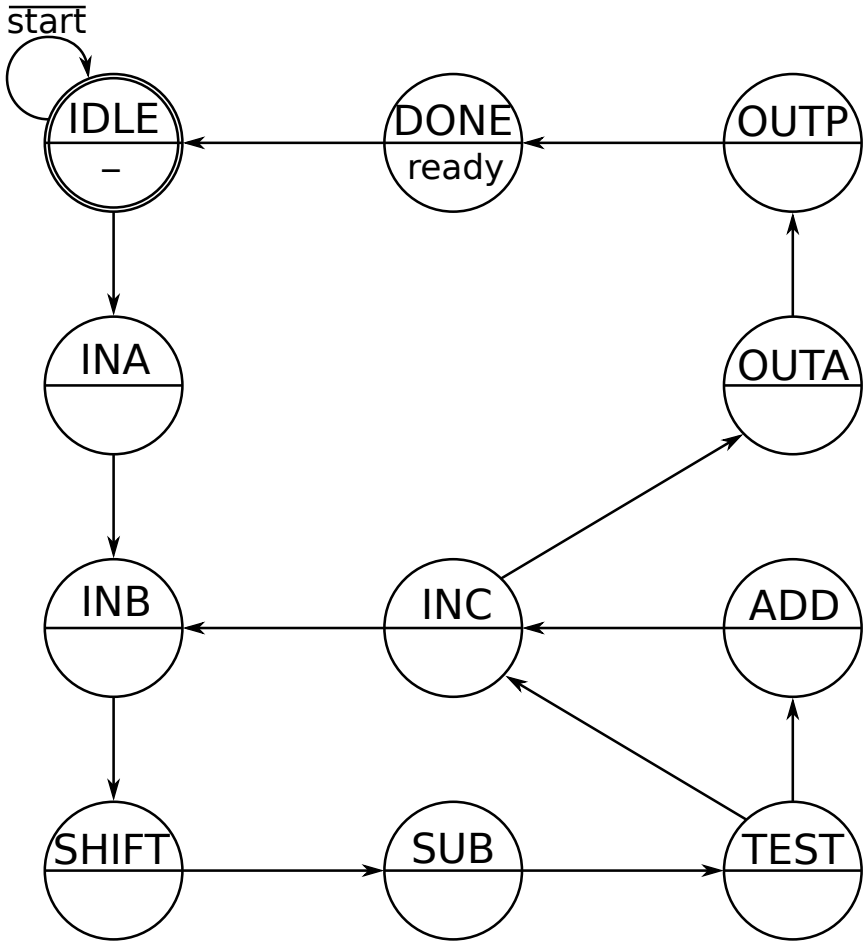


Abbildung 8: **Ersatzautomat, ungültige Lösung streichen!**

Aufgabe 5: (VHDL) **Entfällt für Studium Generale!** 20 Punkte

Abbildung 12 zeigt ein 4-Bit linear rückgekoppeltes Schieberegister (engl. Linear Feedback Shift Register, LFSR) mit den Bit-Positionen Y(3)..Y(0). Bei einem Reset wird das Register auf den Wert "0100" gesetzt. In jedem Takt wird das Register um ein Bit nach rechts geschoben. Das nachzuschiebenne Bit *inter* bekommt den Wert $S_{in} \oplus Y(0) \oplus Y(2)$, wobei $\oplus$ der XOR Operator ist.

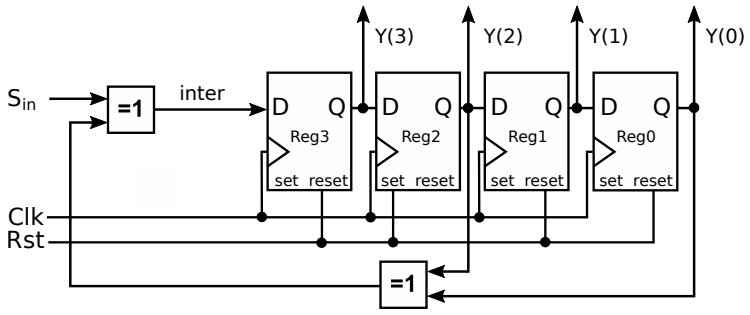
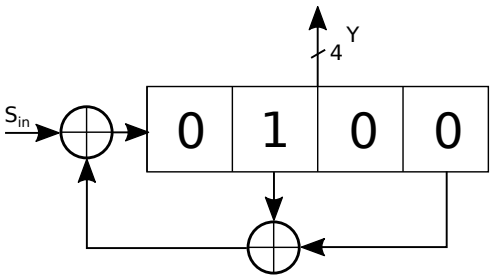


Abbildung 9: 4-Bit linear rückgekoppeltes Schieberegister

a) In Abbildung 10 sehen Sie den Zustand des 4-Bit LFSRs nach einem Reset. Ergänzen Sie in der nebenstehenden Tabelle den Zustand des Registers nach jedem Takt, wenn die Bitfolge 110101_2 bei jeder steigenden Taktflanke, beginnend beim niederwertigsten Bit, nachgeschoben wird. (3 Punkte)



t_0: **Y=**	**„0100“**
t_1: **Y=**	
t_2: **Y=**	
t_3: **Y=**	
t_4: **Y=**	
t_5: **Y=**	
t_6: **Y=**	

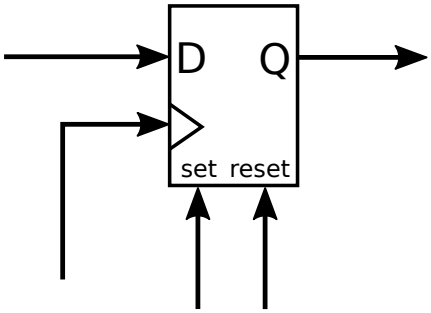
Abbildung 10: Start- und Folgezustände des LFSRs nach t Takten

b) Nachfolgend finden sie ein Code-Fragment für das VHDL Modul LFSR. Ergänzen Sie sowohl die `entity` als auch die `architecture` so, dass das Modul LFSR die Funktionalität der Schaltung aus Abbildung 12 implementiert. (10 Punkte)

Entity LFSR (Ersatz auf Seite 22)

```
library IEEE;
use IEEE.std_logic_1164.ALL;

entity LFSR is
  port(

    Clk ____________________________________________

    Rst ____________________________________________

    Sin ____________________________________________

    Y   ____________________________________________
  );
end LFSR;

architecture Behavioral of LFSR is

  ____________________________________________________________________

  ____________________________________________________________________
begin

  Y <= ____________________________________________

  lfsr_proc : process( ____________________________________ )
  begin

  ____________________________________________________________________

  ____________________________________________________________________

  ____________________________________________________________________

  ____________________________________________________________________

  ____________________________________________________________________

  ____________________________________________________________________

  ____________________________________________________________________

  ____________________________________________________________________

  ____________________________________________________________________

  ____________________________________________________________________

  ____________________________________________________________________
  end process;

  ____________________________________________________________________

  ____________________________________________________________________
end architecture Behavioral;
```

c) Nun soll das oben genannte LFSR mit Hilfe der Komponente *oneBitReg* (Abb. 11) realisiert werden, dass bei einer steigenden Taktflanke den Wert von D speichert und an Q ausgibt. Dazu muss die Komponente 4 mal instanziiert werden. Ergänzen Sie das nachfolgende Code Fragment so, dass die Funktionalität aus Aufgabe **a)** gegeben ist. (7 Punkte)



VHDL Entity

```
entity oneBitReg is
  port (
    clk   : in std_logic;
    set   : in std_logic;
    reset : in std_logic;
    D     : in std_logic;
    Q     : out std_logic);
end;
```

Abbildung 11: 1-Bit-Register

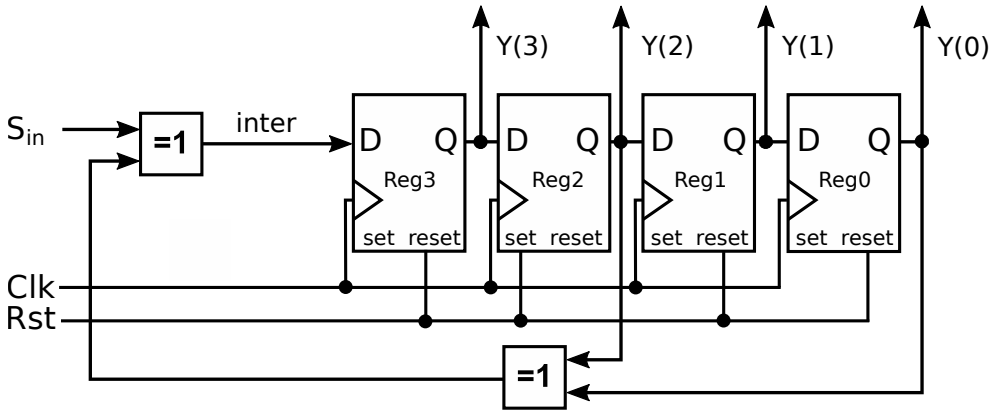


Abbildung 12: 4-Bit linear rückgekoppeltes Schieberegister aus Aufgabenteil a)

VHDL Fragment mit Komponente (Ersatz auf Seite 23)

```
library IEEE; use IEEE.std_logic_1164.ALL;
entity LFSR is
  port(
    Clk ____________________________________________

    Rst ____________________________________________

    Sin ____________________________________________

    Y   ____________________________________________
  );
end LFSR;
architecture Structural of LFSR is
component oneBitReg is
  port (
    clk     ________________________________________

    set     ________________________________________

    reset   ________________________________________

    D       ________________________________________

    Q       ________________________________________
  );
end component;
  signal connector: std_logic_vector(3 downto 0);

  signal ____________________________________________
begin
  inter <= connector(2) xor Sin xor connector(0);
  reg0: oneBitReg port map (clk=>Clk,set=>open, reset=>Rst,
                            D=>connector(1),Q=>connector(0));
  reg1: oneBitReg port map (clk=>Clk,

    ____________________________________________

    ____________________________________________
  );
  reg2: oneBitReg port map (clk=>Clk,

    ____________________________________________

    ____________________________________________
  );
  reg3: oneBitReg port map (clk=>Clk,

    ____________________________________________

    ____________________________________________
  );
  Y <= ____________________________________
end architecture Structural;
```

Entity LFSR (**Ersatz - ungültige Lösung streichen!**)

```
library IEEE;
use IEEE.std_logic_1164.ALL;

entity LFSR is
  port(

    Clk ____________________________________________

    Rst ____________________________________________

    Sin ____________________________________________

    Y   ____________________________________________
  );
end LFSR;

architecture Behavioral of LFSR is

  ____________________________________________________________

  ____________________________________________________________
begin

  Y <= ____________________________________________

  lfsr_proc : process( ____________________________________ )
  begin

  ____________________________________________________________

  ____________________________________________________________

  ____________________________________________________________

  ____________________________________________________________

  ____________________________________________________________

  ____________________________________________________________

  ____________________________________________________________

  ____________________________________________________________

  ____________________________________________________________

  ____________________________________________________________

  ____________________________________________________________
  end process;

  ____________________________________________________________

  ____________________________________________________________
end architecture Behavioral;
```

VHDL Fragment mit Komponente (**Ersatz - ungültige Lösung streichen!**)

```
library IEEE; use IEEE.std_logic_1164.ALL;
entity LFSR is
  port(
    Clk ____________________________________________

    Rst ____________________________________________

    Sin ____________________________________________

    Y   ____________________________________________
  );
end LFSR;
architecture Structural of LFSR is
component oneBitReg is
  port (
    clk     ________________________________________

    set     ________________________________________

    reset   ________________________________________

    D       ________________________________________

    Q       ________________________________________
  );
end component;
  signal connector: std_logic_vector(3 downto 0);

  signal ____________________________________________
begin
  inter <= connector(2) xor Sin xor connector(0);
  reg0: oneBitReg port map (clk=>Clk,set=>open, reset=>Rst,
                            D=>connector(1),Q=>connector(0));
  reg1: oneBitReg port map (clk=>Clk,

    ____________________________________________

    ____________________________________________
  );
  reg2: oneBitReg port map (clk=>Clk,

    ____________________________________________

    ____________________________________________
  );
  reg3: oneBitReg port map (clk=>Clk,

    ____________________________________________

    ____________________________________________
  );
  Y <= ____________________________________
end architecture Structural;
```