Test und Verlässlichkeit Grosse Übung zu Foliensatz 5

Prof. G. Kemnitz

25. November 2016

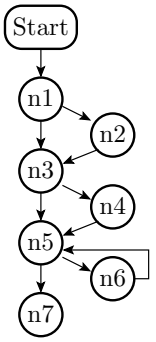
1 Software

Aufgabe 5.1: Kontrollflussgraph

C-Programm zur Berechnung des größten gemeinsamen Teilers.

```
        int ggt(int x, int y) {
         int c;
n1:      if ( x < 0 )
n2:       x = -x;
n3:      if ( y < 0 )
n4:       y = -y;
n5:      while ( y != 0 ) {    // solange y != 0
          // ersetze x durch y und y durch den Rest
          // von x modulo y
n6:       c = x % y; x = y; y = c;
         }
n7:      return x;
        }
```

Zeichnen Sie den Kontrollflussgraph.



Aufgabe 5.2: Kontrollflussgraph 2

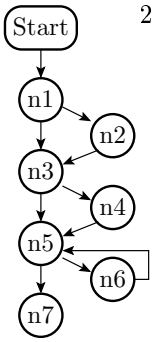
```
    int ggt(int x, int y) {
     int c;
n1:  if (x < 0
n2:   x = -x;
n3:  if (y < 0)
n4:   y = -y;
n5:  while(y != 0){
n6:   c = x % y; x = y; y = c;}
n7:  return x;}
```

Bestimmen Sie den Kontrollflussablauf für das Testbeispiel $x = 4$, $y = 6$. Füllen Sie dazu nachfolgende Tabelle aus:

Anweisung	Wert* von x	Wert* von y
n1	4	6

(* – nach Ausführung der Anweisung).

Zur Kontrolle




```
    int ggt(int x, int y) {
     int c;
n1:   if (x < 0)
n2:    x = -x;
n3:   if (y < 0)
n4:    y = -y;
n5:   while(y != 0){
n6:    c = x % y; x = y; y = c;}
n7:   return x;}
```

Anw.	Wert x	Wert y	Anw.	Wert x	Wert y
n1	4	6	n6	4	2
n3	4	6	n5	4	2
n5	4	6	n6	2	0
n6	6	4	n7	2	0
n5	6	4			

Aufgabe 5.4: Kontrollflussgraph 3

1. Konstruieren Sie ein Testbeispiel, dass jede Anweisung in der Abfolge n1-n2-n3-n4-n5-n6-n7 genau einmal abarbeitet.
2. Konstruieren Sie ein zweites möglichst kurzes Testbeispiel, dass zusätzlich die vom ersten Testbeispiel nicht erfassten Anweisungen n2 und n4 abarbeitet.

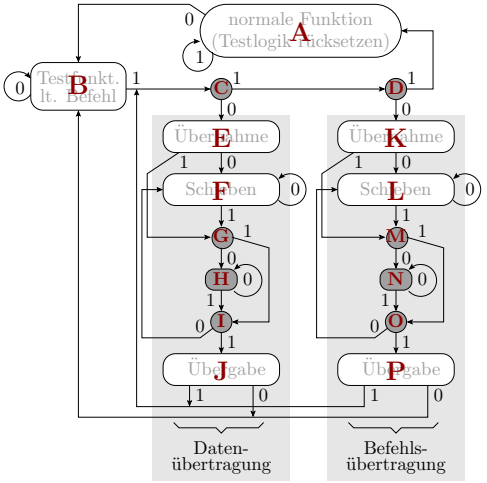
Zur Kontrolle

```
n1:  if (x < 0)
n2:   x = -x;
n3:  if (y < 0)
n4:   y = -y;
n5:  while (y != 0){
n6:   c = x % y; x = y; y = c;}
n7:  return x;
```

1. Für ein Testbeispiel, dass jede Anweisung in der Abfolge n1-n2-n3-n4-n5-n6-n7 genau einmal abarbeitet, muss gelten $x \geq 0$, $y \geq 0$ und $x\%y \Rightarrow 0$, z.B. $x = 4$ und $y = 2$ (Programmablauf siehe folgende Tabelle links).
2. Die Zustände n2 und n4 werden nur bei $x < 0$ und $y < 0$ durchlaufen. Weniger als ein Schleifenduchlauf schließt $y < 0$ aus. Nur ein Schleifendurchlauf verlangt $|x| \,\%\, |y| \Rightarrow 0$, z.B. $x = -9$ und $y = -3$ (Ablauf siehe folgende Tabelle rechts).

Anw.	Wert x	Wert y	Anw.	Wert x	Wert y
n1	4	2	n1	-9	-3
n3	4	2	n2	9	-3
n5	4	2	n3	9	-3
n6	2	0	n4	9	3
n5	2	0	n5	9	3
n7	2	0	n6	3	0
			n5	3	0
			n7	3	0

Aufgabe 5.5: Automatentest TAP-Controlle



Spezifizieren Sie eine Zustandsfolge und die zugehörige TMS-Folge (TMS – Übergangsbedingung an den Kanten), bei der alle 32 Kanten des Automaten mindestens einmal durchlaufen werden.

Die Initialisierung aus einem beliebigen Startzustand erfolgt mit 6×TMS=1.

Zur Kontrolle

Z	?	?	?	?	?	?	A	A	B	B	C	E	F	F	G	H	H	I	J
TMS	1	1	1	1	1	1	1	0	0	1	0	0	0	1	0	0	1	1	0
Z	B	C	E	G	I	E	G	I	J	C	D	K	L	L	M	N	N	O	P
TMS	1	0	1	1	0	1	1	1	0	1	0	0	0	1	0	0	1	1	0
Z	B	C	D	K	M	O	L	M	O	P	C	D	A						
TMS	1	1	0	1	1	0	1	1	1	1	1	1							

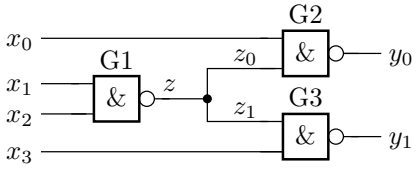
(farbig – wiederholt durchlaufene Kanten).

2 Schaltkreistest

2.1 Haftfehler

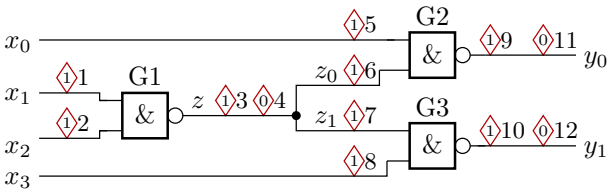
Aufgabe 5.6: Haftfehlermenge

Stellen Sie für die nachfolgende Schaltung die Menge aller Haftfehler auf. Bei mehreren identisch nachweisbaren Haftfehlern ist jeweils nur einer in die Fehlermenge aufzunehmen.



Hinweis: Notation der Haftfehler sa0(<Signalname>) bzw. sa1(<Signalname>).

Zur Kontrolle

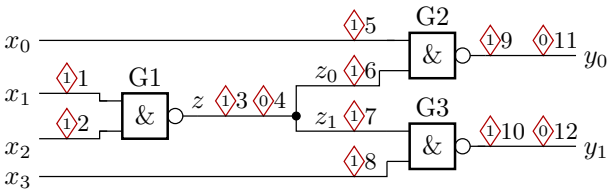


Liste der Modellfehler:

Nr.	0	1	2	3	4	5
Fehler	sa1(x0)	sa1(x1)	sa1(x2)	sa1(x3)	sa0(z)	sa1(z)
Nr.	6	7	8	9	10	11
Fehler	sa1(z0)	sa1(z1)	sa0(y0)	sa1(y0)	sa0(y1)	sa1(y1)

Aufgabe 5.7: Fehlersimulation

Schreiben Sie ein C-Programm zur fehlerparallelen Simulation der Schaltung aus der Aufgabe zuvor. Gutsimulation in Bit 0, Simulation der Fehler in den den Fehlern zugeordneten Bits 1 bis 12:



Programmrahmen:

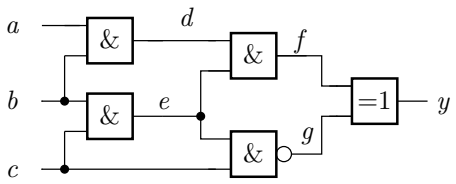
```
uint16_t x0, x1, x2, x3, z, z0, z1, y;
<wiederhole für alle 16 Eingabemöglichkeiten>{
 <Simulation der Gatter und Fehler>
}
```

Zur Kontrolle

```
uint16_t x0=0, x1=0, x2=0, x3=0, z, y;
for (x3=0;x3==0xFF;~x3){
 for (x2=0;x2==0xFF;~x2){
  for (x1=0;x1==0xFF;~x1){
   for (x0=0;x0==0xFF;~x0){
    x0 = x0 | 1<<1;   // F1: sa1(x0)
    x1 = x1 | 1<<2;   // F2: sa1(x1)
    x2 = x2 | 1<<3;   // F3: sa1(x2)
    x3 = x3 | 1<<4;   // F4: sa1(x3)
    z  = ~(x1 & x2); // Gatter G1
    z  = z & ~(1<<5);// F5: sa0(z)
    z  = z | 1<<6;    // F6: sa1(z)
    z0 = z; z1 = z;   // Auffächerung
    z0 = z0 | 1<<7;   // F7: sa1(z0)
    z1 = z1 | 1<<8;     // F8: sa1(z1)
    y0 = ~(x0 & z0);    // Gatter G2
    y0 = y0 & ~(1<<9); // F9: sa0(y0)
    y0 = y0 | 1<<10;    // F10: sa1(y0)
    y1 = ~(x0 & z0);    // Gatter G3
    y1 = y1 & ~(1<<11);// F11: sa0(y1)
    y1 = y1 | 1<<12;    // F12: sa1(y1)
   }
  }
 }
```

2.2 Testberechnung

Aufgabe 5.8: D-Algorithmus[1]

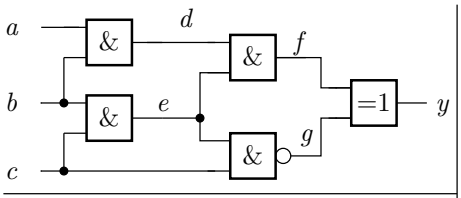


1. Geben Sie alle Möglichkeiten für die Sensibilisierung eines D-Pfads von Eingang c zum Ausgang y an.
2. Suchen Sie für den Haftfehler sa1(a) einen Test mit dem D-Pfad $a \rightarrow d \rightarrow f \rightarrow y$.

Kennzeichnung der Wertefestlegungen: F – lokale Fehlernachweisbedingung; I – implizite Festlegung; E – Entscheidung; $\bar{E}$ – invertierte Entscheidung; W – Widerspruch.

[1] Aus http://www.eda.ei.tum.de/fileadmin/tueieda/www/EI-BSc/EDS/tutorium/Tutorial_Dalgorithmus.pdf

Zur Kontrolle

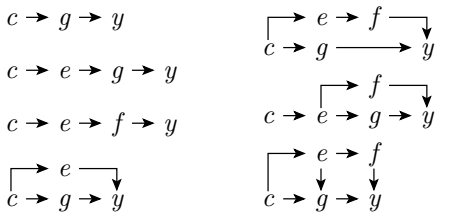


Aufgabenteil 2: Testsuche für sa1(a), D-Pfad $a - d - f - y$

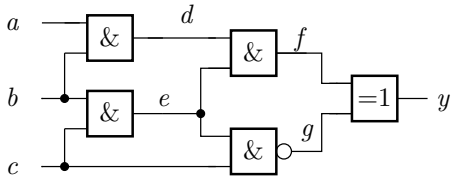
a =D	F
$b = 1$	I
d =D	I
$e = 1$	I
f =D	I
$c = 1$	I
$g = 0$	I
d =D	I

$(a, b, c) = (1, 1, 1)$

Aufgabenteil 1: alle Pfade



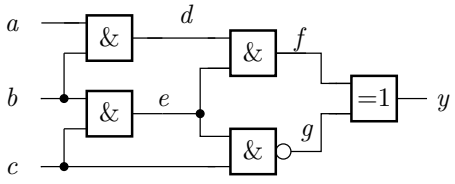
Aufgabe 5.9: D-Algorithmus Fortsetzung



3. Suchen Sie für den Haftfehler sa0(e) einen Test mit dem D-Pfad $e \to g \to y$.
4. Suchen Sie für den Haftfehler sa0(c) einen Test mit dem D-Pfad $c \to e \to f \to y$.

Kennzeichnung der Wertefestlegungen: F – lokale Fehlernachweisbedingung; I – implizite Festlegung; E – Entscheidung; $\bar{\text{E}}$ – invertierte Entscheidung; W – Widerspruch.

Zur Kontrolle



Aufgabenteil 3: sa1(e) über D-Pfad $e - g - y$

e =D	F
$b = 1$	I
$c = 1$	I
$g = \bar{\text{D}}$	I
$f = 1$	E
$d = 1$	I
$e = 0$	W

→

$f = 0$	$\bar{\text{E}}$
$y = \bar{\text{D}}$	I

$(a, b, c) = (*, 1, 1)$

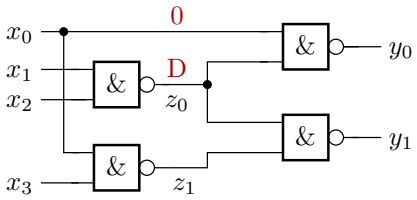
Aufgabenteil 4: sa0(c) über D-Pfad $c - e - f - y$

$c = \bar{\text{D}}$	F
$b = 1$	I
$e = \bar{\text{D}}$	I
g =D	I
$d = 1$	I
$f = \bar{\text{D}}$	I
$y = 1$	W

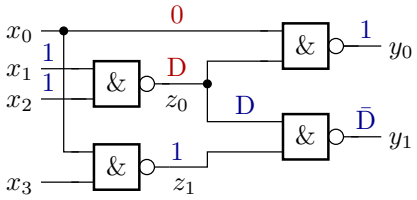
keine Lsg.

Aufgabe 5.10: Implikationstest

Bestimmen Sie für die nachfolgende Schaltung mit den beiden Signalfestlegungen (einmal »0« und einmal »D«) alle damit implizit festgelegten Signalwerte.



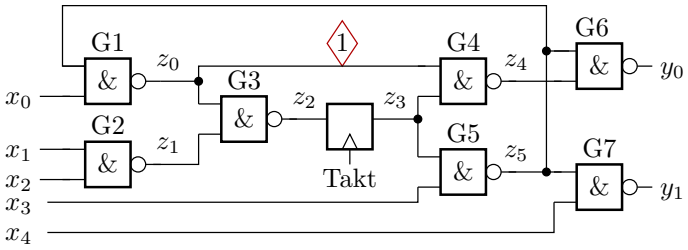
Zur Kontrolle



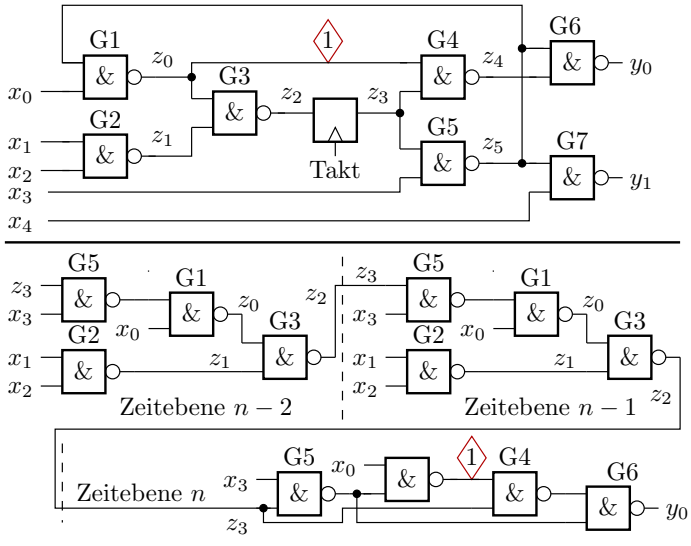
- Die vorgegebene »0« verbietet Weiterführung des D-Pfads nach y_0.
- Eindeutiger D-Pfad nach y_1.
- $z_0 = \mathrm{D}$ impliziert $x_1 = x_2 = 1$.
- $x_0 = 0$ impliziert $y_0 = 1$ und $z_1 = 1$

Aufgabe 5.11: Kombinatorische Ersatzschaltung

Rollen Sie die nachfolgende Schaltung zu einer kombinatorischen Ersatzschaltung für die Testberechnung des eingezeichneten Haftfehlers auf mit einer Begrenzung der Länge der Steuerspfade auf max. drei Zeitebenen (max. 3 Schaltungskopien).



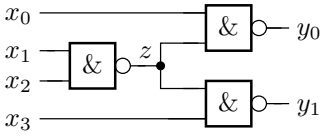
Zur Kontrolle



2.3 Andere Fehlermodelle

Aufgabe 5.12: Toogle-Überdeckung

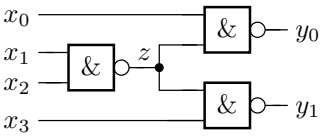
1. Kontrollieren Sie für die nachfolgende Schaltung und den angegebenen Testsatz, welche Signale noch nicht mindestens einmal den Wert null und einmal den Wert eins annehmen.
2. Erweitern Sie den Testsatz um zusätzliche Eingaben so, dass eine 100%ige Toggle-Test-Überdeckung erzielt wird.



x_3	x_2	x_1	x_0	z	y_1	y_0
0	1	1	0			
1	1	0	0			
1	1	1	0			

Hinweis: Verwenden Sie als Hilfmittel die Wertetabelle rechts.

Zur Kontrolle

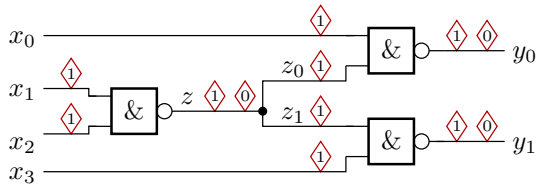


x_3	x_2	x_1	x_0	z	y_1	y_0
0	1	1	0	0	1	1
1	1	0	0	1	0	1
1	1	1	0	1	0	1
1	0	0	1	1	0	0

1. Mit den ersten drei Testbeispielen sind nicht abgedeckt: $x_0 = 2$, $x_2 = 0$ und $y_0 = 0$
2. Für 100% Toogle-Überdeckung genügt ein weiterer Test:

$$x_3x_2x_1x_0 = 1001$$

Aufgabe 5.14: Haftfehlerüberdeckung Toogle-Test

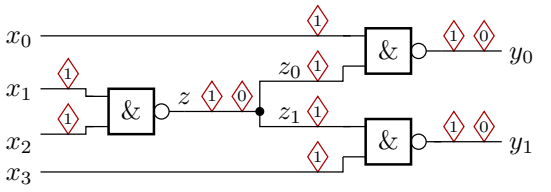


x_3	x_2	x_1	x_0	z	y_1	y_0	sa1(x_0)	sa1(x_1)	sa1(x_2)	sa1(x_3)	sa1(z)	sa0(z)	sa1(z_0)	sa1(z_1)	sa1(y_0)	sa0(y_0)	sa1(y_1)	sa0(y_1)
0	1	1	0															
1	1	0	0															
1	1	1	0															

1. Bestimmen Sie für die Eingaben des Toggle-Tests aus der Aufgabe zuvor, welche der eingezeichneten Haftfehler nachgewiesen werden.
2. Wie groß ist die Haftfehlerüberdeckung des Toggle-Tests?

Zur Kontrolle

1. Kennzeichnung der nachweisbaren Haftfehler:



x_3	x_2	x_1	x_0	z	y_1	y_0	sa1(x_0)	sa1(x_1)	sa1(x_2)	sa1(x_3)	sa1(z)	sa0(z)	sa1(z_0)	sa1(z_1)	sa1(y_0)	sa0(y_0)	sa1(y_1)	sa0(y_1)
0	1	1	0	0	1	1										×		×
1	1	0	0	1	0	1	×	×		×		×				×	×	
1	1	1	0	0	1	1				×	×			×		×	×	
1	0	0	1	1	0	0	×	×	×	×		×	×	×	×		×	

2. Haftfehlerüberdeckung: 100%

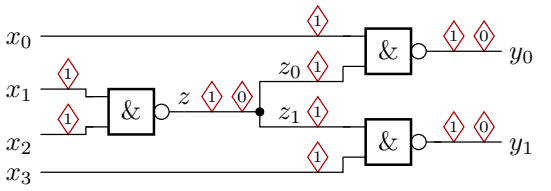
Aufgabe 5.15: Gatterverzögerungsfehler

1. Für welche der Haftfehler in der Aufgabe zuvor wird der korrespondierende Gatterverzögerungsfehler nachgewiesen.
2. Wie groß ist die Gatterverzögerungsfehlerüberdeckung des betrachteten Toggle-Tests?

Hinweis: Der korrespondierende Gatterverzögerungsfehler zu »stuck-at-0« ist »slow-to-rise« und zu »stuck-at-1« »slow-to-fall«. Zusätzliche Nachweisbedingung ist ein Signalwechsel am Fehlerort.

Zur Kontrolle

1. Nachweisbare Verzögerungsfehler farbig unterlegt:



zusätzliche Änderung des Wertes am Fehlerort

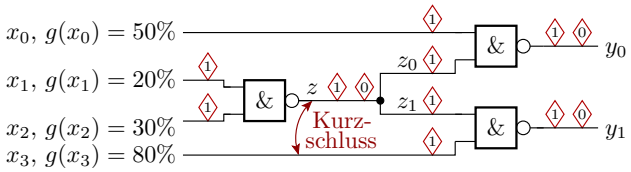
x_3	x_2	x_1	x_0	z	y_1	y_0	sa1(x_0)	sa1(x_1)	sa1(x_2)	sa1(x_3)	sa1(z)	sa0(z)	sa1(z_0)	sa1(z_1)	sa1(y_0)	sa0(y_0)	sa1(y_1)	sa0(y_1)
0	1	1	0	0	1	1										×		×
1	1	0	0	1	0	1	×	×		×		×				×	×	
1	1	1	0	0	1	1				×	×			×		×	×	
1	0	0	1	1	0	0	×	×	×	×		×	×	×	×		×	

(sa0 ⇒ »slow-to-rise«; sa1 ⇒ »slow-to-fall«)

2. Überdeckung für Verzögerungsfehler: $\frac{10}{12}$ ≈83%

Aufgabe 5.16: Kurzschlussnachweis

Bei dem eingezeichneten Kurzschluss soll sich null durchsetzen. Welche der eingezeichneten Haftfehler teilen sich Nachweisbedingungen mit dem Kurzschluss und wie groß ist für jeden dieser Haftfehler die Wahrscheinlichkeit, dass wenn ein Test ihn nachweist, auch der Kurzschluss nachgewiesen wird?



Hinweis: $g(...)$ – Signalwichtungen, Auftrittshäufigkeit einer Eins. Die bedingte Wahrscheinlichkeit, dass ein Kurzschluss von einem Haftfehlertest nachgewiesen wird, ist die Wichtung $g(...)$ der anderen beteiligten Leitung oder deren Gegenwahrscheinlichkeit.

Zur Kontrolle

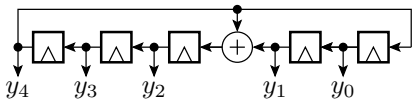
Der Kurzschluss »0 setzt sich durch« teilt sich mit den Haftfehlern sa0(x_3) und sa0(z) Anregungs- und Beobachtungsbedingungen. Bedingte Wahrscheinlichkeiten für den Kurzschlussnachweis, wenn einer dieser Haftfehler nachgewiesen wird:

- sa0(x_3), wenn $z = 0$. $P(z = 0) = g(x_1) \cdot g(x_2) = 6\%$
- sa0(z), wenn $x_3 = 0$. $P(x_3 = 0) = 1 - g(x_3) = 20\%$

2.4 Selbsttest

Aufgabe 5.17: LFSR-Zykluslänge

Untersuchen Sie für das nachfolgende 5-Bit linear rückgekoppelte Schieberegister die Zyklusstruktur.



1. Bestimmen Sie für jeden der 32 möglichen Zustände alle erreichbaren Folgezustände.
2. Wie viele unterschiedliche Testeingaben lassen sich maximal hintereinander erzeugen?
3. Wie lautet die zyklisch generierte Testeingabefolge, wenn der Generator mit $y_4y_3y_2y_1y_0 = 01011$ initialisiert und jeder zweite Zustand als Testeingabe verwendet wird?

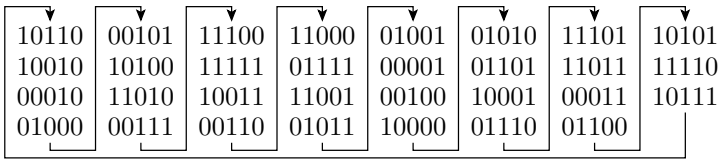
Zur Kontrolle

1. Zyklusstruktur:

Nullzyklus: 00000 | Zyklus der anderen Zustände Länge: $2^r - 1 = 31$

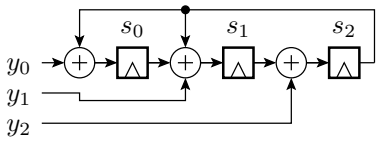
00001	00101	10001	11111	01100	11001
00010	01010	00111	11011	11000	10111
00100	10100	01110	10011	10101	01011
01000	01101	11100	00011	01111	10110
10000	11010	11101	00110	11110	01001
					10010

2. Maximale Zykluslänge: 31
3. Jede zweite Zustand ab 01011:



Aufgabe 5.18: Signaturregister

Gegeben ist das nachfolgende Sinaturregister, eine Sollausgabefolge von einem fehlerfreien Testobjekt und die verfälsche Ausgabefolge eines fehlerhaften Testobjekts:

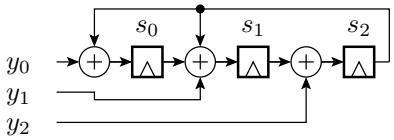


korrekte Folge			fehlerhafte F.		
y_0	y_1	y_2	y_0	y_1	y_2
1	0	1	1	1	1
0	0	1	0	0	1
1	1	0	0	1	0
0	0	0	0	0	0

1. Wie groß ist die Wahrscheinlichkeit, dass das Signaturregister für eine verfälschte Datenfolge ein anderes Prüfkennzeichen als für die korrekte Datenfolge berechnet?
2. Welche Signatur (Prüfkennzeichen) hat die Sollausgabefolge und welche die fehlerhafte Ausgabefolge?
3. Kontrollieren Sie am Beispiel, dass die Signatur der Differenzfolge gleich der Differenz der Signaturen ist (Überlagerungssatz).

Zur Kontrolle

1. Maskierungswahrscheinlichkeit: $p_M = 2^{-3} = 0{,}125$
2. Signatur der Soll- und der Fehlerfolge:

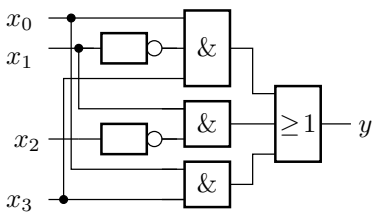


	korrekte Folge			Sollsignatur			fehlerhafte F.			Fehlersign.		
Schritt	y_0	y_1	y_2	s_0	s_1	s_2	y_0	y_1	y_2	s_0	s_1	s_2
0	1	0	1	0	0	0	1	1	1	0	0	0
1	0	0	1	1	0	1	0	0	1	1	1	1
2	1	1	0	0	1	1	0	1	0	0	1	0
3	0	0	0	0	0	1	0	0	0	0	1	1
Endw.				1	1	0				1	1	1

3. Differenzen rot unterlegt.

Aufgabe 5.19: Selbsttest

Testobjekt sei die nachfolgende Schaltung:



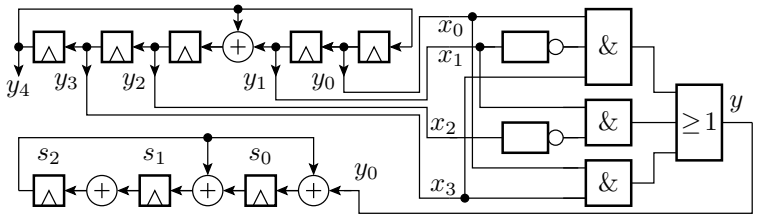
	$x_3x_2x_1x_0$	y	$s_2s_1s_0$
1	0 0 1 0 1		0 0 0
2			
3			
4			
5			
Signatur:			

1. Schließen Sie am Eingang das 5-Bit LSFR von Seite 9 mit y_0 an x_0, y_1 an x_1 bis y_3 an x_3 und an den Ausgang das 3-Bit-Signaturregister von Seite 10 an mit y an y_0 an.
2. Bestimmen Sie die Sollsignaltur für Generatorstartwert 00101, den Signaturregisterstartwert 000 und 5 Testschritte. Füllen Sie dazu rechts die Tabelle aus.

Zur Kontrolle

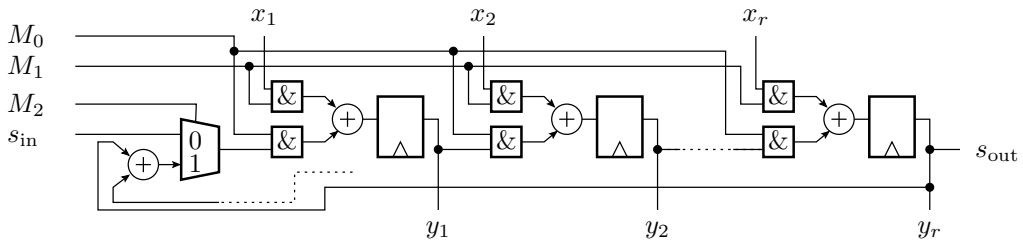
1. Selbstestlösung mit Pseudo-Zufallsgenerator am Eingang und Signatoranalysator am Ausgang (Bild unten).
2. Wertetabelle des Testaublaufs (Bild rechts).

	$x_3x_2x_1x_0$	y	$s_2s_1s_0$
1	0 0 1 0 1	0	0 0 0
2	0 1 0 1 0	1	0 0 0
3	1 0 1 0 0	0	0 0 1
4	0 1 1 0 1	1	0 1 0
5	1 1 0 1 0	1	1 0 1
Signatur:			0 0 1



Aufgabe 5.20: Built-in Logic Block Observer

Die gezeigte Schaltung ist ein **B**uilt-**I**n **L**ogic **B**lock **O**bserver (BILBO) und führt in Abhängigkeit von den Steuersignalen M_0 bis M_2 eine der Funktionen aus: Intialisierung, normales Register, Schieberegister, Pseudo-Zufallsgenerator oder Signaturegister.



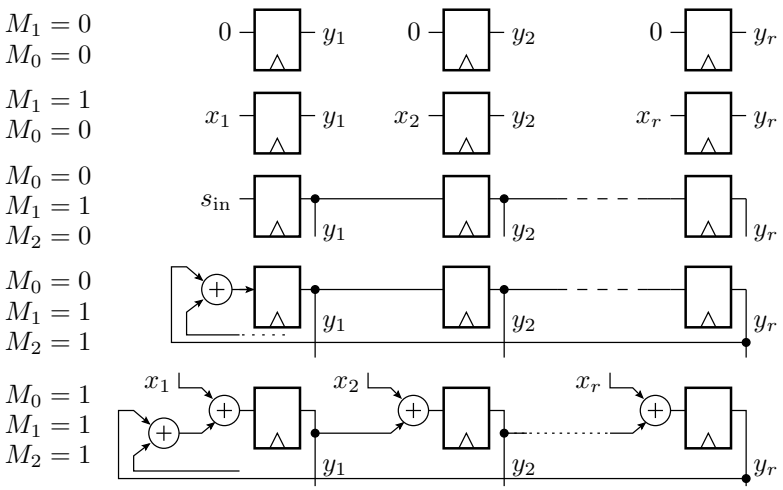
1. Welche Steuersignalbelegung steuert welche Funktion?
2. Vereinfachen Sie für jede dieser Steuersignalbelegungen die Schaltung durch Konstanteneliminierung (Vereinfachen bzw. Weglassen der logischen Verknüpfungen mit Konstanten.)

Zur Kontrolle

1. Steuersignalbelegungen und Funktion:

M_2	M_1	M_0	Funktion
-	0	0	Rücksetzen: $y_i = 0$
0	0	1	Schieben: $y_0 = s_{\text{in}}$, sonst $y_i = y_{i-1}$
-	1	0	Register: $y_i = x_i$
1	0	1	LFSR: $y_0 = y_{\text{r}} \oplus \ldots$, sonst $y_i = y_{i-1}$
1	1	1	Signaturregister: $y_0 = x_0 \oplus y_{\text{r}} \oplus \ldots$, sonst $y_i = x_i \oplus y_{i-1}$

2. Vereinfachte Schaltung für jede dieser Steuersignalbelegungen:


$M_1 = 0$
$M_0 = 0$
$M_1 = 1$
$M_0 = 0$
$M_0 = 0$
$M_1 = 1$
$M_2 = 0$
$M_0 = 0$
$M_1 = 1$
$M_2 = 1$
$M_0 = 1$
$M_1 = 1$
$M_2 = 1$
0
y_1
0
y_2
0
y_r
x_1
y_1
x_2
y_2
x_r
y_r
s_{in}
y_1
y_2
y_r
y_1
y_2
y_r
x_1
x_2
x_r
y_1
y_2
y_r