RHEINISCH-
WESTFÄLISCHE
TECHNISCHE
HOCHSCHULE
AACHEN

LEHRSTUHL
INFORMATIK XI
SOFTWARE FÜR
EINGEBETTETE
SYSTEME

Professor Dr.-Ing. Stefan Kowalewski
Hilal Diab, M.Sc.
Kamal Barakat, M.Sc
Dipl.-Inform. Dominik Franke

Aachen, 15. Januar 2010
SWS: V4/Ü2, ECTS: 7

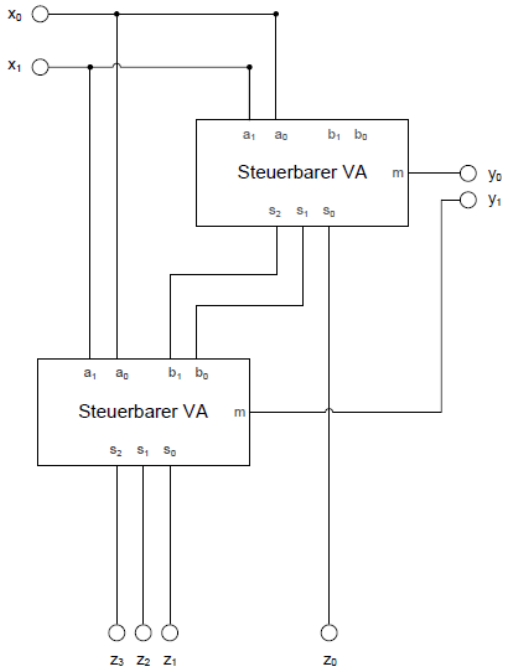
Einführung in die Technische Informatik

WS 2009/2010

Blatt 10: Musterlï¿½sung

Aufgabe 1: Digitale Schaltungen

Betrachten Sie die gegebene Schaltung



Hinweis: Die beiden steuerbaren Volladdierer besitzen zwei Eingänge (a_1, a_0) und (b_1, b_0), die entsprechend der Vorlesung addiert werden, falls der Steuereingang $m = 1$ ist. Eingang (a_1, a_0) wird abgeschaltet (d.h. $a_1 = a_0 = 0$), sobald der Steuereingang $m = 0$ ist. Das Ergebnis wird jeweils in (s_2, s_1, s_0) geschrieben.

a) Stellen Sie die Wertetabelle für die Schaltfunktion $F((x_1, x_0), (y_1, y_0)) = (z_3, z_2, z_1, z_0)$ auf.

b) Welche Funktion wird durch F realisiert?

c) Formulieren Sie mit eigenen Worten die Funktionsweise der Schaltung.

Aufgabe 2: Addierwerke

a) Demonstrieren Sie die Arbeitsweise eines Parallel-Addierwerks, eines Serien-Addierwerks und eines von-Neumann-Addierwerks (jeweils 4-Bit-Addierwerke) für die nacheinander ausgeführten Berechnungen 4+10, 13+5 und 10+10, indem Sie die Inhalte vorhandener Delays schrittweise in einer Tabelle protokollieren.[1]

S	U	$P_3P_2P_1P_0$	dezimal	$A_3A_2A_1A_0$	dezimal(inkl. U)
0	0	0000	0	0000	0
1	0	0100	4	1010	10
...	...	....	...	...	...

b) Nennen Sie für jedes der drei Addierwerke mindestens einen Vor- und einen Nachteil.

c) Beschreiben Sie die Funktionsweise eines Ripple-Carry-Adders und berechnen Sie die Schaltzeit. Gehen Sie hierfür von einem 12-Bit Addierwerk aus und nehmen Sie eine hypothetische Schaltzeit von 10 psec ($10^{-11}s$) pro Gatter an. Erklären Sie, wie mit Hilfe von Carry-Bypass-Addiernetzen die Dauer der Berechnung verkürzt werden kann und stellen Sie eine Beispielrechnung auf.

Aufgabe 3: Subtraktion zweier Boolescher Zahlen

In dieser Aufgabe sollen Sie Gatterschaltungen für einen Halbsubtrahierer und einen Vollsubtrahierer entwerfen. Ein Halbsubtrahierer subtrahiert zwei Dualziffern und erzeugt deren Differenz und ein Unterlauf-Bit. Wenn der Minuend kleiner ist als der Subtrahend, dann wird das Unterlauf-Bit auf 1 gesetzt, andernfalls auf 0. Ein Vollsubtrahierer besitzt drei Eingaben (Dualziffern) und zwei einstellige Ausgaben. Zwei der Eingaben sind die zu subtrahierenden Ziffern, die dritte Eingabe ist der Unterlauf der niederwertigen Bitposition. Die Ausgaben bestehen aus Differenz und neuem Unterlauf.

a) Stellen Sie Funktionstafeln für beide Bausteine auf. Hierzu ist es u.U. hilfreich, die schriftliche Subtraktion zweier Dezimalzahlen zu betrachten und dieses Verfahren auf die Subtraktion zweier Dualzahlen zu übertragen.

b) Leiten Sie für die Ausgaben beider Bausteine Boolesche Funktionen mit Hilfe von $\cdot, +, ^-$ her.

c) Vereinfachen Sie die Booleschen Funktionen für den Vollsubtrahierer mit Hilfe eines Karnaugh-Diagramms.

d) Zeichnen Sie Schaltnetze für Halb- und Vollsubtrahierer. Nutzen Sie hierbei nur AND-, OR- und NOT-Gatter.

Lösungsvorschlag

[1] Das Status-Delay S entfällt bei Parallel- und Serien-Addierwerk.

a) Betrachten wir zunächst die Subtraktion bei Dezimalzahlen, z.B.:

Minuend	51
Subtrahend	29
Unterlauf	1
Ergebnis	22

Ist an der betrachteten Stelle der Subtrahend kleiner oder gleich dem Minuenden, so ist die entsprechende Stelle im Ergebnis direkt durch die Differenz gegeben; es tritt kein Unterlauf auf. Ist der Subtrahend hingegen größer, so tritt ein Unterlauf auf und es wird das Ergebnis von 10+Minuend-Subtrahend ermittelt. Im Beispiel: 9 ist größer als 1, also ergibt sich 10 + 1 - 9 = 2. Unterläufe werden stets zum Subtrahenden der nächsthöheren Stelle addiert.
Ubertragen wir dieses Prinzip auf Dualzahlen, so sieht eine Subtraktion wie folgt aus:

Minuend	110011
Subtrahend	011101
Unterlauf	111
Ergebnis	010110

Wiederum gilt, dass wenn der Subtrahend größer ist als der Minuend, addieren wir 10 (die Dualdarstellung der Basis 2) auf den Minuenden und vermerken einen Unterlauf. Allgemein erhalten wir folgene Funktionstabellen für Halb- und Vollsubtrahierer, wobei m den Minuenden, s den Subtrahenden, R das Ergebnis und U den Unterlauf bezeichnet. u steht für den Unterlauf der niederwertigen Bitposition.:

Funktionstabelle für den Halbsubtrahierer(HS):

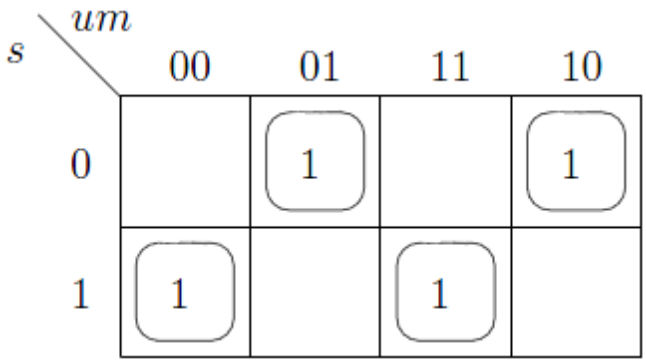
m	s	R	U
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Funktionstabelle für den Vollsubtrahierer(VS):

m	s	u	R	U
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

b)
$$
\begin{aligned}
R_{HS} &= \bar{m}s + m\bar{s} \\
U_{HS} &= \bar{m}s \\
R_{VS} &= \bar{u}\bar{m}s + \bar{u}m\bar{s} + u\bar{m}\bar{s} + ums \\
U_{VS} &= \bar{u}\bar{m}s + u\bar{m}\bar{s} + u\bar{m}s + ums
\end{aligned}
$$

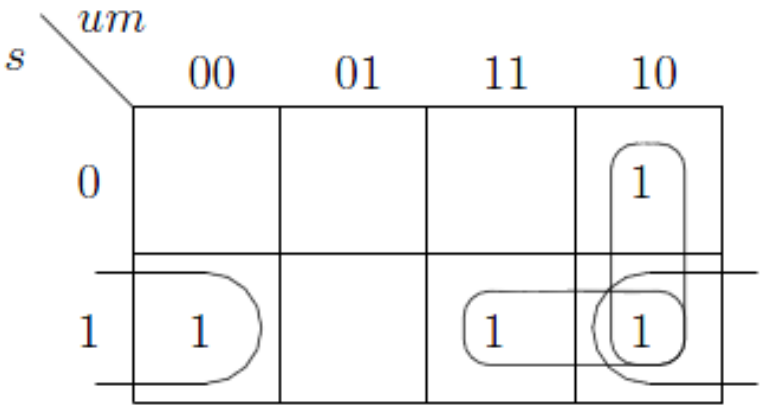
c) Für R erhalten wir:



$$R_{VS} = \bar{u}\bar{m}s + \bar{u}m\bar{s} + u\bar{m}\bar{s} + ums$$

(Die Funktion läßt sich also nicht weiter minimieren.)

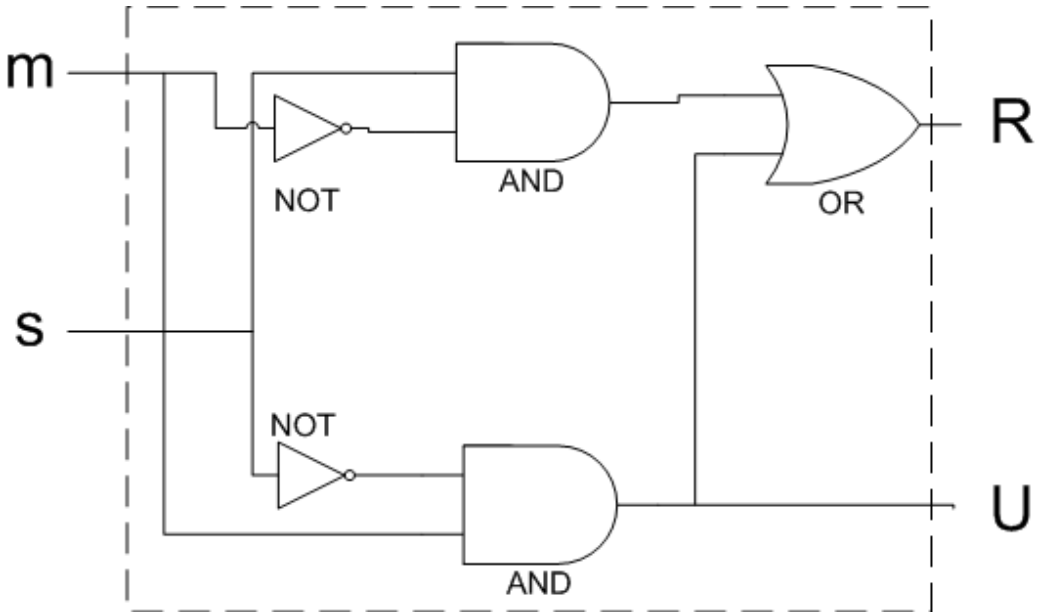
Für U ergibt sich:



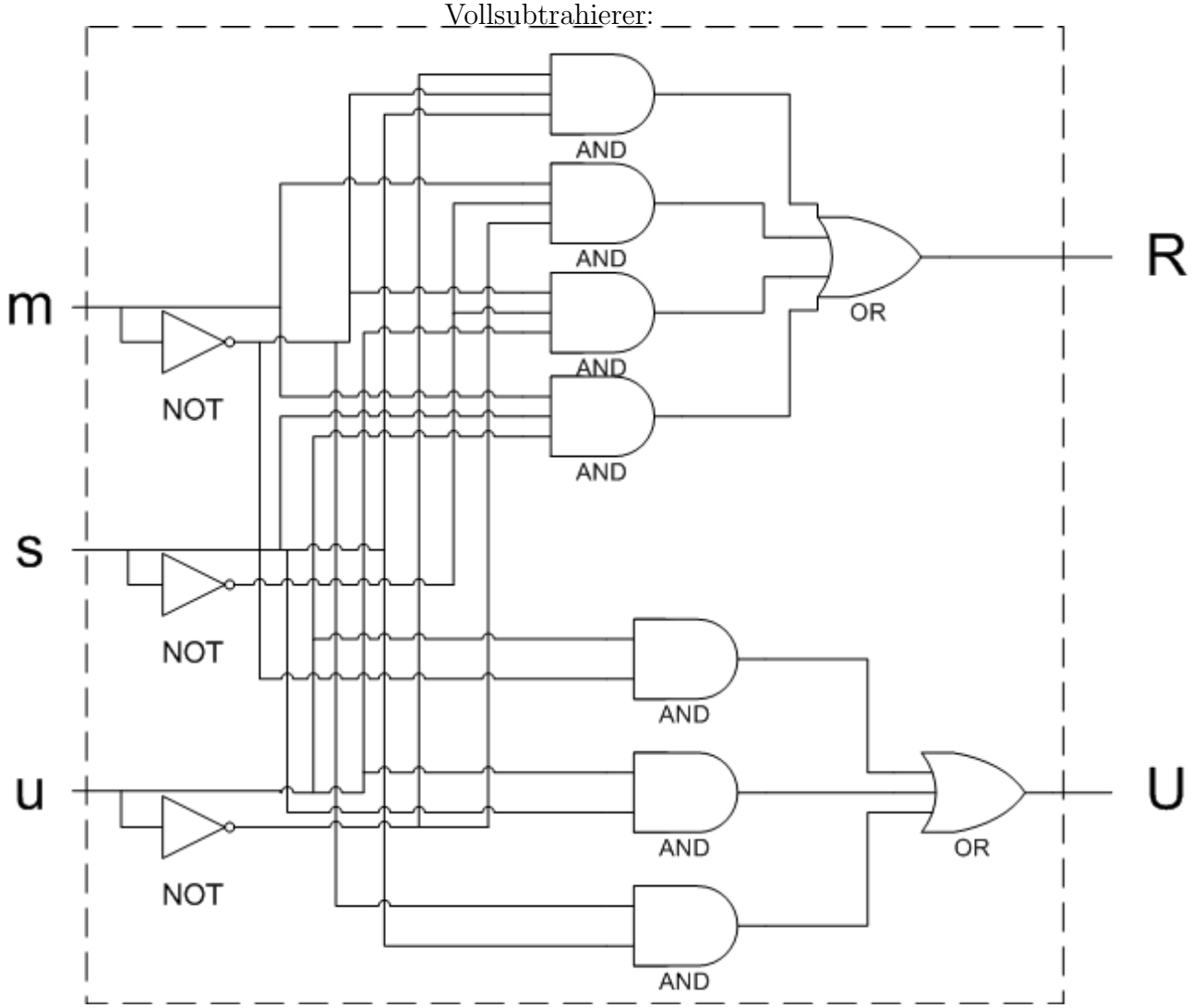
$$U_{VS} = u\bar{m} + us + \bar{m}s$$

d) Halb- und Vollsubtrahierer kann man darstellen wie folgt:

Halbsubtrahierer:

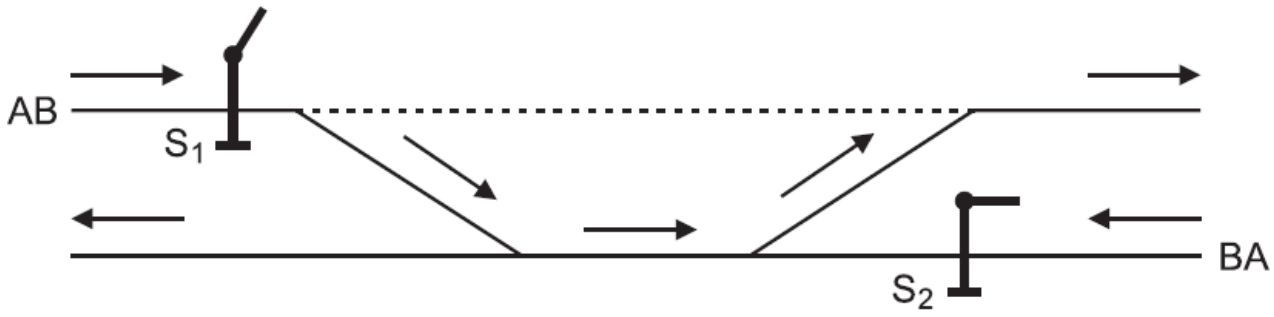


Vollsubtrahierer:



Aufgabe 4: (⋆)Schaltungsentwurf

Zwischen zwei Orten A und B gibt es eine Schienenverbindung mit zwei Gleiswegen AB und BA. Dabei nutzen von A nach B fahrende Züge den Gleisweg AB, während von B kommende Züge den Gleisweg BA befahren. Bauarbeiten auf einem Teilabschnitt von AB führen jedoch zu einer Ausnahmesituation, die in der folgenden Abbildung illustriert ist:



Danach müssen AB befahrende Züge für einen kurzen Teilabschnitt auf den Gleisweg BA ausweichen. Der Zugverkehr auf diesem Teilabschnitt soll durch zwei einfache Signalanlagen S1 und S2 geregelt werden, die entweder 'freie Fahrt' oder 'unbedingter Halt' signalisieren können. Weiterhin sollte die Regelung derart erfolgen, dass die beiden Signalanlagen in ihren Signalen als Paar (S_1, S_2) aufgefasst stets in einem Zyklus die Zustände $(0,0), (1,0), (0,0), (0,1)$ und wieder $(0,0)$ durchlaufen, wobei 'freie Fahrt' mit 1 und 'unbedingter Halt' mit 0 codiert ist.

a) Entwerfen Sie ein optimiertes (= minimiertes) (Steuer-)Schaltwerk, das die beiden Signalanlagen in der beschriebenen Weise und unter Verwendung der angegebenen Codierung steuert.

b) Demonstrieren Sie die Arbeitsweise Ihres Steuerwerks, indem Sie die Inhalte aller vorhandenen Delays vor und nach jedem Takt für einen kompletten Zyklus protokollieren.

c) Erweitern Sie Ihr Steuerwerk um eine Steuerleitung Z, deren Wert darüber bestimmt, ob der regelnde Betrieb stattfindet oder ob die beiden Signalanlagen ständig auf 'unbedingter Halt' gesetzt sind.

   Hinweis: Dies lässt sich mit zwei zusätzlichen Gattern realisieren lassen.

Lösungsvorschlag

a) Die Schwierigkeit der Aufgabe besteht darin, dass ein aktuelles Signalpaar (0, 0) in Abh ängigkeit von der vorherigen Signalstellung in der sich anschließenden Signaländerung sowohl in das Signalpaar (1, 0) als auch in das Signalpaar (0, 1) überführt werden kann bzw. muss.

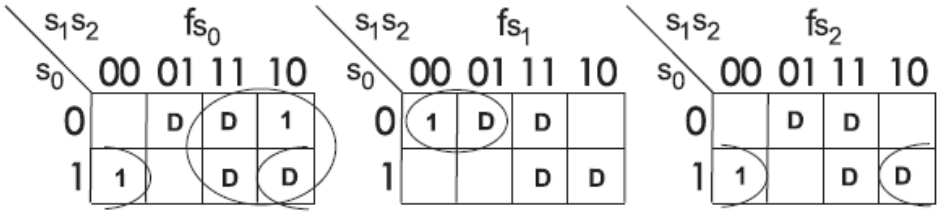
Man könnte 'intern' z. B. das zweite (0, 0)-Paar durch (1, 1) ersetzen und mit einer Art Ringzähler arbeiten, der in einem Zyklus $(0,0) \rightarrow (1,0) \rightarrow (1,1) \rightarrow (0,1) \rightarrow (0,0) \rightarrow ...$ 'zählt'. Eine nachgeschaltete 'Umwandlungslogik' muss dann das Paar (1, 1) in (0, 0) umrechnen, während die anderen Zählerausgaben durchgeleitet werden. Ein Kritikpunkt an dieser Lösung im Hinblick auf die Aufgabenstellung ist, dass sie die beiden Signalanlagen nicht direkt in der beschriebenen Weise und unter Verwendung der angegebenen Codierung steuert, sondern die Zustandscodierung von den Steuersignalen trennt. (Das hier drei der vier Zustandscodierungen mit den Steuersignalen übereinstimmen können, ändert nichts an der hier prinzipiell anderen Vorgehensweise.) Im Allgemeinen muss dann auch bedacht werden, dass die 'Umwandlungslogik' evtl. Hasard-frei sein sollte, um sicheren Signalbetrieb zu gewährleisten. (In der Vorlesung sind Hasards nicht besprochen worden.)

Wir nutzen zur Lösung eine Schaltfunktion $f : B^3 \rightarrow B^3$, die über ein zusätzliches Bit beide ' (0, 0)-Varianten' unterscheiden kann. Dabei wird f nur partiell genutzt. Das zusätzliche Bit entspricht dann einem ' internen' Steuersignal S_0. Wir notieren S_0^{neu} , S_1^{neu} , S_2^{neu} als Funktion f von S_0^{alt} , S_1^{alt} , S_2^{alt}. Damit ergibt sich die unten folgende Wertetabelle, in der wir die ausgezeichneten Signalpaare (S_1, S_2) des Zyklus mit Z_0, Z_1, Z_2, Z_3 bezeichnen. Die Funktionswerte S_1^{neu} und S_2^{neu} sind durch den einzuhaltenden Zyklus vorgegeben. Allerdings sind zur Festlegung von S_0^{neu} teilweise auch Alternativen möglich, d. h. es gibt mehrere Lösungen, die bis auf ' symmetrische' Vertauschungen identisch sind.

'altes' Signalpaar	S_0^{alt}	S_1^{alt}	S_2^{alt}	S_0^{neu}	S_1^{neu}	S_2^{neu}	'neues' Signalpaar
Z_0	0	0	0	0	1	0	Z_1
tritt niemals auf	0	0	1	D	D	D	
Z_1	0	1	0	1	0	0	Z_2
tritt niemals auf	0	1	1	D	D	D	
Z_2	1	0	0	1	0	1	Z_3
Z_3	1	0	1	0	0	0	Z_0
tritt niemals auf	1	1	0	D	D	D	
tritt niemals auf	1	1	1	D	D	D	

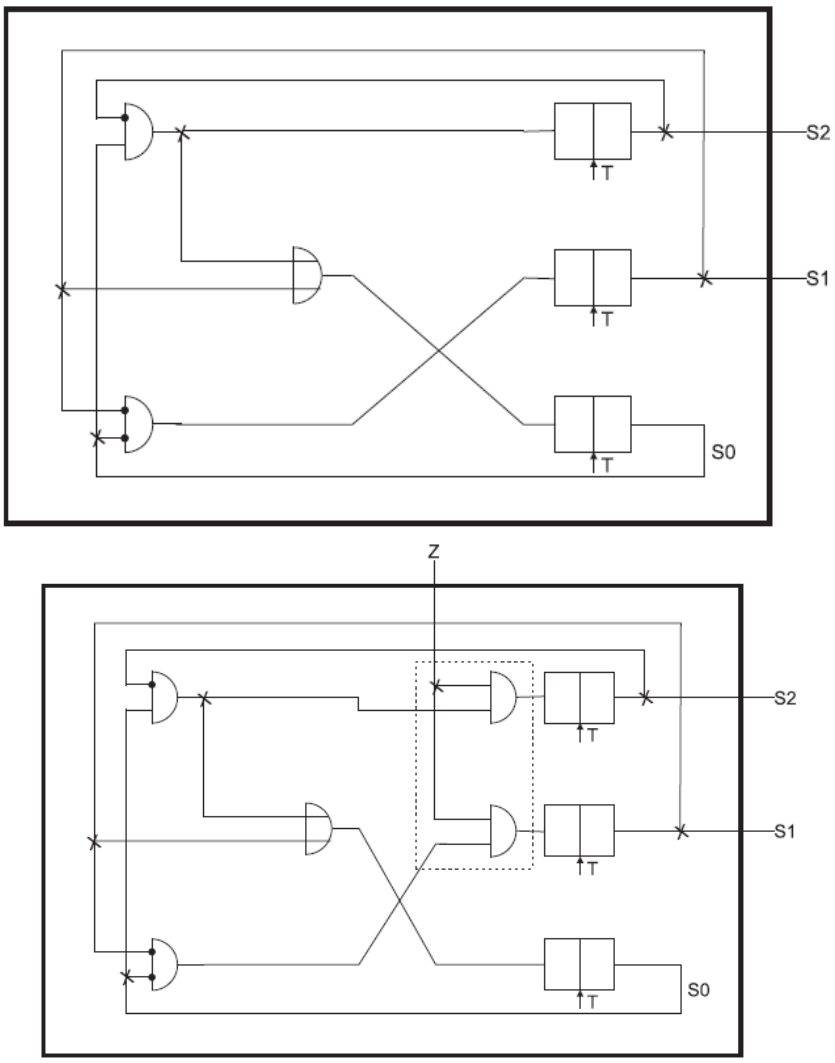
Mit Hilfe von Karnaugh-Diagrammen und unter Ausnutzung der Don't-Care-Werte ergeben sich für die drei Booleschen Funktionen S_0^{neu} , S_1^{neu} , S_2^{neu} der gesuchten Schaltfunktion f die folgenden Minimalpolynome:

$S_0^{neu} = S_0^{alt} \cdot \bar{S}_2^{alt} + S_1^{alt} \quad S_1^{neu} = \bar{S}_0^{alt} \cdot \bar{S}_1^{alt} \quad S_2^{neu} = S_0^{alt} \cdot \bar{S}_2^{alt}$



Damit ergibt sich das gesuchte Schaltwerk unter Verwendung von 3 Delay-Bausteinen

wie in der folgenden Abbildung:



b) Wir notieren $|S_0|S_1S_2|$ und starten initial mit $S_0 = 0$ sowie $S_1 = S_2 = 0$, d.h. beide Signalanlagen signalisieren 'unbedingter Halt'.
$|0|00| \overset{Takt}{\longrightarrow} |0|10| \overset{Takt}{\longrightarrow} |1|00| \overset{Takt}{\longrightarrow} |1|01| \overset{Takt}{\longrightarrow} |0|00| \overset{Takt}{\longrightarrow} ...$

c) Siehe letzte Abbildung Aufgabenteil a)

Aufgabe 5: ($\star$)Addierwerke

Demonstrieren Sie die Arbeitsweise eines Parallel-Addierwerks, eines Serien-Addierwerks und eines von-Neumann-Addierwerks (jeweils 3-Bit-Addierwerke) für die nacheinander ausgeführten Berechnungen 3+5, 7+2, 1+3, indem Sie die Inhalte vorhandener Delays schrittweise in einer Tabelle protokollieren.

S	U	$P_2P_1P_0$	dezimal	$A_2A_1A_0$	dezimal(inkl. U)
0	0	000	0	000	0
...	...	....	...	...	...

Lösungsvorschlag

- Lösungsvorschlag Parallel-Addierwerk:

3+5:

U	$P_2P_1P_0$	dezimal	$A_2A_1A_0$	dezimal(inkl. U)
0	000	0	000	0
0	011	3	101	5
1	000	0	000	8

7+2:

U	$P_2P_1P_0$	dezimal	$A_2A_1A_0$	dezimal(inkl. U)
0	000	0	000	0
0	111	7	010	2
1	000	0	001	9

1+3:

U	$P_2P_1P_0$	dezimal	$A_2A_1A_0$	dezimal(inkl. U)
0	000	0	000	0
0	001	1	011	3
0	000	0	100	4

- Lösungsvorschlag Serien-Addierwerk:

3+5:

U	$P_2P_1P_0$	dezimal	$A_2A_1A_0$	dezimal(inkl. U)
0	000	0	000	0
0	011	3	101	5
1	001	1	010	10
1	000	0	001	9
1	000	0	000	8

7+2:

U	$P_2P_1P_0$	dezimal	$A_2A_1A_0$	dezimal(inkl. U)
0	000	0	000	0
0	111	7	010	2
0	011	7	101	5
1	001	1	010	10
1	000	0	001	9

1+3:

U	$P_2P_1P_0$	dezimal	$A_2A_1A_0$	dezimal(inkl. U)
0	000	0	000	0
0	001	1	011	3
1	000	0	001	9
1	000	0	000	8
0	000	0	100	4

- Lösungsvorschlag Von-Neumann-Addierwerk:

3+5:

S	U	$P_2P_1P_0$	dezimal	$A_2A_1A_0$	dezimal(inkl. U)
0	0	000	0	000	0
1	0	011	3	101	5
1	0	010	2	110	6
1	0	100	4	100	4
0	1	000	0	000	8

7+2:

S	U	$P_2P_1P_0$	dezimal	$A_2A_1A_0$	dezimal(inkl. U)
0	0	000	0	000	0
1	0	111	7	010	2
1	0	100	4	101	5
0	1	000	0	001	9

1+3:

S	U	$P_2P_1P_0$	dezimal	$A_2A_1A_0$	dezimal(inkl. U)
0	0	000	0	000	0
1	0	001	1	011	3
1	0	010	2	010	2
1	0	100	4	000	0
0	0	000	0	100	4

Aufgabe 6: ($\star$)Latches

Diese Aufgabe ist eine Erweiterung zu Aufgabe 6, Übungsblatt 9.

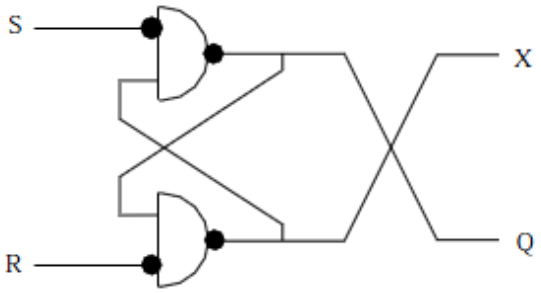
a) Zeichnen Sie Zustandsdiagramme (genau so wie auf Übungsblatt 9, Aufgabe 6) für ein

- AND-Latch
- NAND-Latch
- XOR-Latch

(d.h., ersetzen Sie beide Gatter im Latch entsprechend durch ANDs, NANDs, XORs).

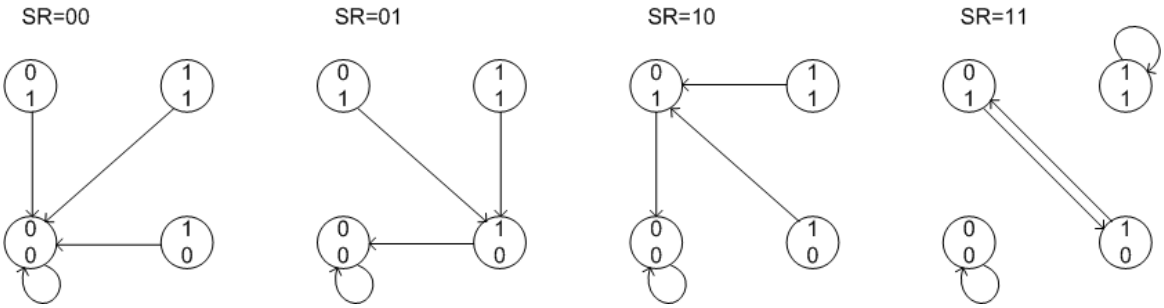
b) Geben Sie für jedes in Aufgabenteil a) behandelte Latch eine entsprechende Beschreibung (ähnlich wie auf Übungsblatt 9, Aufgabe 6) und begründen Sie, ob diese Schaltung als 1-Bit-Speicherbaustein dienen könnte oder nicht.

c) Zeichnen Sie die Zustandsdiagramme für die Schaltung aus der nachfolgenden Abbildung und vergleichen Sie deren Funktionalität mit der Funktionalität des NOR-Latches.

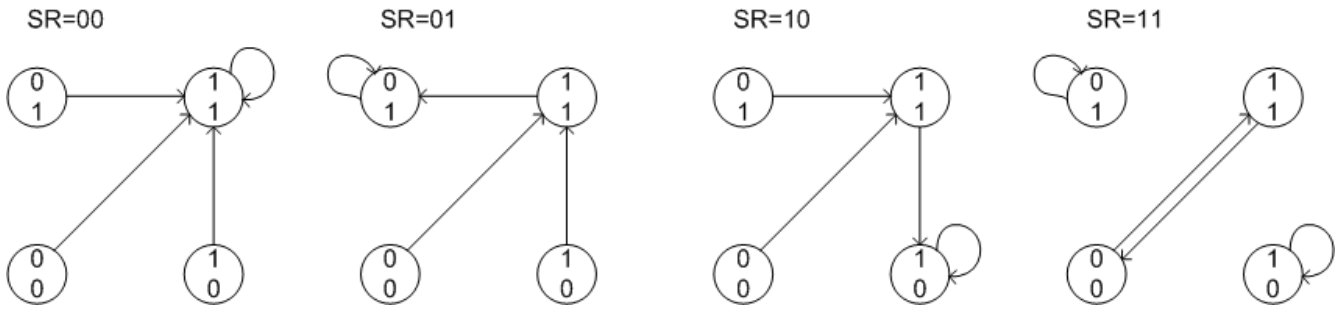


Lösungsvorschlag

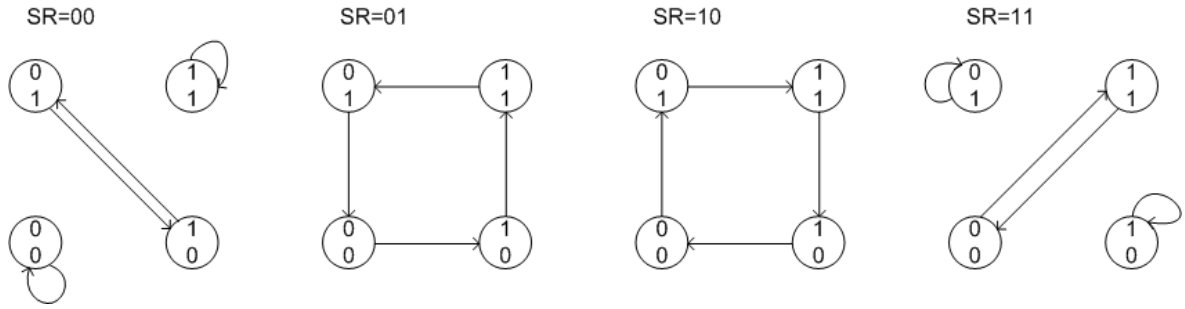
a) AND-Latch:



NAND-Latch:



XOR-Latch:

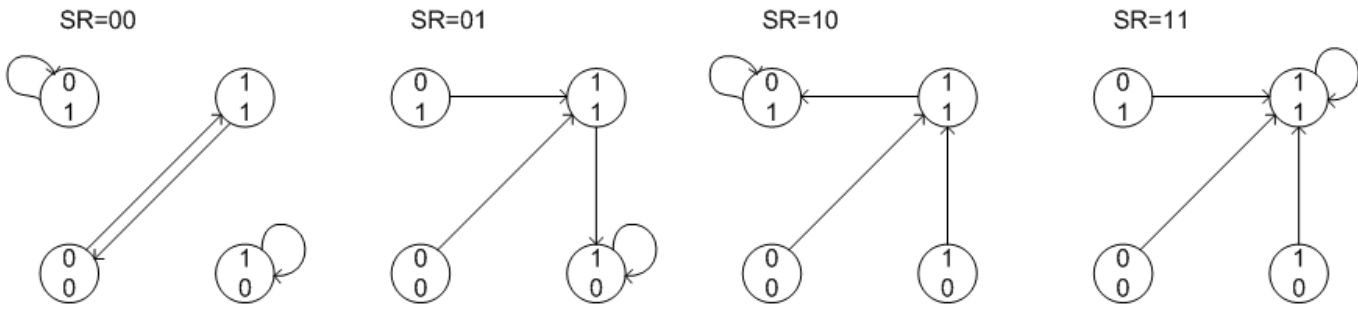


b) AND-Latch:
Das AND-Latch ist nicht geeignet als 1-Bit Speicher, da es bei SR≠11 in den Fangzustand 00 und bei SR=11 ggf. in einen instabilen Zustand kommt. Damit ist die Forderung nicht erfüllt, die Information zu erhalten.

NAND-Latch:
Das NAND-Latch kommt für SR≠11 immer in einen stabilen Zustand.
Für SR=00 bleibt das Latch im Zustand $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$.
Für SR=01 bleibt das Latch im Zustand $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$.
Für SR=10 bleibt das Latch im Zustand $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$.
Alle anderen Fälle bringen keine Funktionalität(insbesondere SR=11).
Somit ist die Forderung erfüllt die Information zu erhalten. Beim NAND-Latch sind die Eingangspegel HIGH und LOW vertauscht, d.h. für die Eingänge S,R gilt HIGH=0 und LOW=1. Für die Ausgangspegel gilt weiterhin HIGH=1 und LOW=0.
Für z.B. SR=01=HIGH LOW gilt am Ausgang des Latches Q=1=HIGH und $\bar{Q}$=0=LOW. Das Latch befindet sich im Zustand 'setzen'.

XOR-Latch:
Das XOR-Latch ist als 1-Bit Speicher nicht geeignet, da das Latch immer einen instabilen Zustand erreichen kann.

c) Zustandsdiagramm:



Dieses Latch verhält sich änlich zum NAND-Latch, jedoch gilt für die Eingangspegel aufgrund der Negation vor S und R: HIGH=1 und LOW=0.
Das NOR-Latch und dieses Latch füren zum gleichen Zustand für gleiche Eingangspegel. Allerdings ist die Zustandsänderung beim NOR-Latch anders als bei diesem Latch. Für SR=10 geht das NOR-Latch vom Zustand $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ zuerst in den Zustand $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$ und anschließend in den Zustand $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ über. Für SR=10 geht dieses Latch jedoch vom Zustand $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ zuerst in den Zustand $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ und anschließend in den Zustand $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ über.

Aufgabe 7: Matlab/Simulink

Benutzen Sie für diese Aufgabe die Datei 'uebung10.mdl', die im L2P unter Lernmaterialien->Matlab zur Verfügung gestellt wird.

a) Konstruieren Sie zwei Halbaddierer.

   Hinweis: Konstruieren Sie zuerst einen Halbaddierer, ziehen Sie einen Rahmen um alle Bauteile und wählen Sie dann rechte Maustaste->Create Subsystem. Duplizieren Sie jetzt das entstandene Subsystem.

b) Konstruieren Sie einen Volladdierer mit Hilfe der zwei Halbaddierer aus Aufgabenteil a).

c) Erstellen Sie nun ein 4-Bit Paralleladdierwerk mit Hilfe des Volladdiererbausteins aus Aufgabenteil b).

d) Sie sollen jetzt einen 3-Bit up-Counter entwerfen, also einen Zähler, der immer wieder von 0 aufwärts zählt. Geben Sie zunächst ein Zustandsdiagramm für den Counter an.

e) Konstruieren Sie den Counter mit Hilfe von D-Flip-Flops.

   Hinweis: Achten Sie unbedingt darauf, dass am !CLR Eingang des D-Flip-Flops der Signalpegel HIGH=1 anliegt, da das Flip-Flop seinen Inhalt sonst löschen wird. Benutzen Sie hierfür den Simulinkbaustein 'Constant', der in der Datei 'uebung10.mdl' zur Verfügung gestellt wird, und setzen Sie die Eigenschaft 'Constant value' des Bausteins auf 1.

f) Mit Flip-Flops können Speicherbausteine realisiert werden. Ein solcher Speicherbaustein, der in CPU's verwendet wird, ist das Register. Ein einfaches Register hat die Kommandos 'Laden' und 'Lesen'. Entwerfen Sie solch ein Register mit einer Speichergröße von vier Bits. Es soll vier Eingangsleitungen, vier Ausgangsleitungen und einen Eingang für das clock Signal besitzen. Weiterhin soll es zwei Steuerleitungen haben (Load, Read), um die zwei Kommandos auszuführen. Das Verhalten des Registers wird mit folgender Tabelle veranschaulicht:

LOAD	READ	Eingangsleitungen	Registerinhalt	Ausgangsleitungen
LOW	LOW	1011	0100	0000
LOW	HIGH	1011	0100	0100
HIGH	LOW	1011	1011	0000
HIGH	HIGH	1011	1011	1011

Für die Pegel gilt HIGH=1 und LOW=0.

Lösungsvorschlag
Up-Counter Zustandsdiagramm: