HEINRICH-HERTZ-INSTITUT FÜR SCHWINGUNGSFORSCHUNG
BERLIN-CHARLOTTENBURG

Technischer Bericht Nr. 139

Die Erzeugung einschrittiger Kettencodes

von

Dipl.-Ing. Klaus Böttcher

Berlin
1971

Technischer Bericht Nr. 139

Die Erzeugung einschrittiger Kettencodes

Zusammenfassung:

Im ersten Teil des Berichtes werden die Eigenschaften eines einschrittigen Kettencodes beschrieben.

Weiterhin wird ein Verfahren angegeben, nach welchem mit Hilfe eines Suchprogramms einschrittige Kettencodes mit einem Minimalabstand von $d \geq 3$ zwischen Codewörtern, deren Abstand im Codealphabet ≥ 3 ist, erzeugt werden können.

Im zweiten Teil des Berichtes wird gezeigt, wie ein Dualcode in einen solchen einschrittigen Kettencode, der eine Fehlerkorrektur erlaubt, umgesetzt werden kann.

Der Bearbeiter

K. Böttcher

(Dipl.-Ing. K.Böttcher)

Der Abteilungsleiter:

(Prof.Dr.-Ing.Erich R.Berger)

Der Institutsdirektor:

P. Matthieu

(Prof.Dr.phil.P.Matthieu)

Berlin-Charlottenburg, den 23. August 1971

Inhaltsverzeichnis

1. Grundlagen über Codes mit Nebenbedingungen /1/

1.1 Einschrittige Codes

Unter einschrittigen Codes versteht man solche Codes, bei denen sich die im Codealphabet aufeinanderfolgenden Codewörter jeweils nur in einer Binärstelle unterscheiden. Ein Beispiel dafür ist der Gray-Code, der aber für den Übergang von 9 auf 0 nicht mehr einschrittig ist. Diesen Nachteil vermeidet der Code von Glixon.

Einschrittige Codes finden Verwendung in Analog-Digital-Umsetzern (ADU) z.B. mechanischen Winkelcodierern. Da sich von Codewort zu Codewort jeweils nur eine Stelle ändert, tritt auch bei Zwischenstellungen der Codescheibe höchstens ein Fehler von einer Amplitudenstufe auf.

1.2 Kettencodes

Einen Kettencode kann man erzeugen, wenn man die N(n-stelligen) Codewörter so anordnet, daß sie eine in sich geschlossene Kette der Länge N bilden. Die einzelnen Codewörter erhält man dann durch schrittweises Verschieben eines Ablesefensters der Länge n auf der Kette (Codespur). Als Beispiel sei hier ein Code mit dem Umfang N=16 und n=4 angegeben (Abb.1).

Ein solcher Kettencode kann auch durch ein rückgekoppeltes Schieberegister erzeugt werden. Dabei ist es je nach Anordnung der Rückführungen möglich, einen Code mit allen möglichen Wörtern (mit Ausnahme des Nullwortes, im Beispiel 0000) oder Codes mit verkürztem Umfang zu erzeugen.

Verwendet man einen solchen Code in einem ADU, so bedeutet das, daß man nur noch eine Codespur braucht.

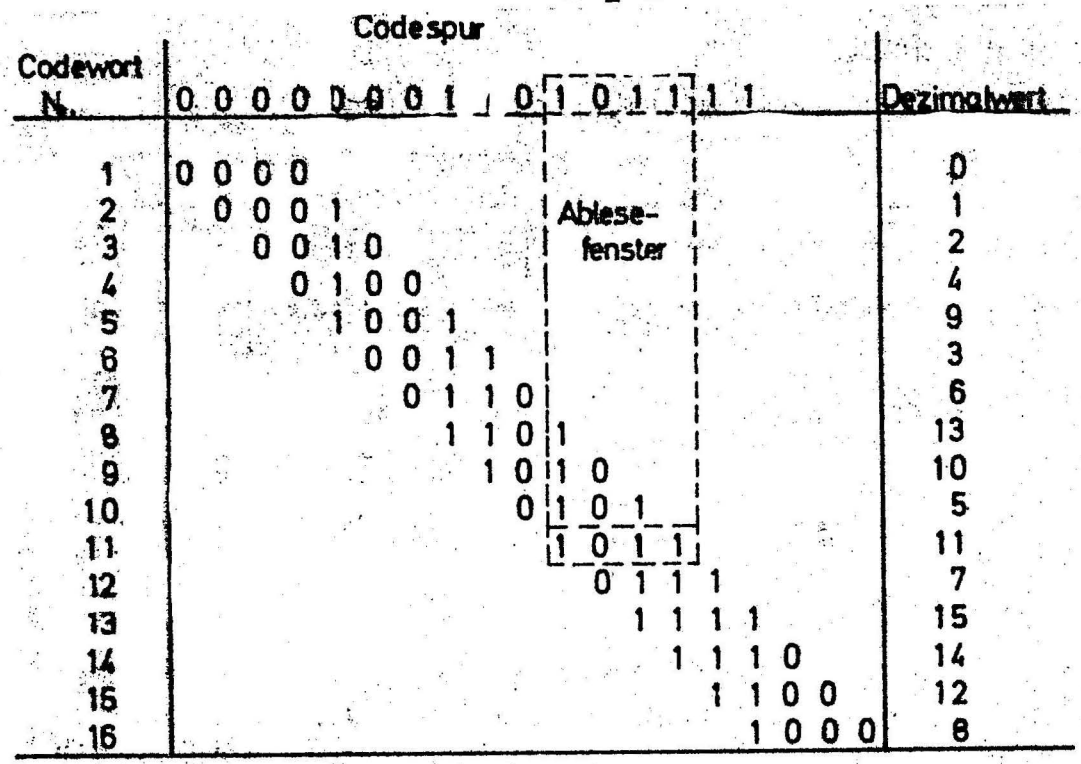


Abb.: 1 Kettencode

1.3. Einschrittige Kettencodes

Vereinigt man die Eigenschaften der beiden bisher beschriebenen Codetypen, so erhält man einen einschrittigen Kettencode. Solche Codes wurden von TOOTILL /2/ untersucht. Für die Aufstellung dieser Codes existieren notwendige Bedingungen:

Der Codeumfang N ist stets gerade. N/2 muß ganzzahlig teilbar sein durch die Stellenzahl n. Ferner muß N stets kleiner sein als 2^n.

Als neuer Begriff wird die Blocklänge eingeführt. Sie gibt an, wie oft zusammenhängende, mit "1" belegte Abschnitte in der Codespur (Kette) vorkommen. Diese Blocklänge m ist der Quotient aus N/2n. Die Anzahl der mit "0" belegten Blöcke ist ebenfalls gleich m. Die Abtastung der Codespur erfolgt durch Lesestationen. Für ihre Verteilung längs der Spur gilt folgendes: um die Aufstellung des Codes zu erleichtern, soll der Abstand r der Lesestationen gleich sein; für eine gerade Anzahl von Blöcken sei der Abstand r=2m, d.h., die Lesestationen sind gleichmäßig über den ganzen Umfang der Codespur verteilt; für ungerades m ist ebenfalls r=2m möglich, daneben aber auch r=m, d.h. die Lesestationen sind nur über den halben Umfang der Codespur verteilt.

Nach TOOTILL erfolgt die Aufstellung eines einschrittigen Kettencodes durch Probieren.

Man sucht die ersten r Codewörter. Alle anderen sind zyklische Permutationen. Falls r=2m gewählt wurde, darf der Code das "1-Wort" und das "0-Wort" nicht enthalten, da bei der zyklischen Permutation jedes der beiden in sich selbst übergeht. Wurde r=m gewählt, so kehrt die letzte Stelle invertiert zurück. Deshalb können die unter dieser Bedingung erzeugten Codes "0"-Wort und "1"-Wort enthalten. Hat man auf diese Weise ein vollständiges Codealphabet erhalten, so muß man sich noch überzeugen, ob alle Codewörter verschieden sind.

Die folgenden drei Beispiele (Abb. 2a,b,c) sollen die bisherigen Ausführungen ergänzen. Die Codes a) und b) wurden von TOOTILL angegeben. Dabei sind die Anzahlen der "0" und "1" im Code gleich. Daß dieses nicht notwendig ist, zeigt das Beispiel c).

Das Codealphabet erhält man, wenn man die Codespur schrittweise an den Lesestationen vorbeischiebt.

Erzeugt man Kettencodes auf die von TOOTILL beschriebene Weise, so kann man keine Aussagen über den Minimalabstand zwischen weiter voneinander entfernten Codewörtern machen.

1.4 Einschrittige Kettencodes aus Zyklen

Wie schon oben erwähnt, entsteht ein einschrittiger Kettencode aus r Codewörtern und deren zyklischen Permutationen. Im folgenden wird nur der Fall für r=2m, d.h. die letzte Stelle kehrt unverändert wieder, behandelt.

Für die weitere Behandlung des Problems sind einige Vorbetrachtungen über Zyklen notwendig. Jede Binärfolge soll durch die natürliche Zahl gekennzeichnet werden, die der dualen Darstellung dieser Binärfolge entspricht. Beschränkt man sich auf eine

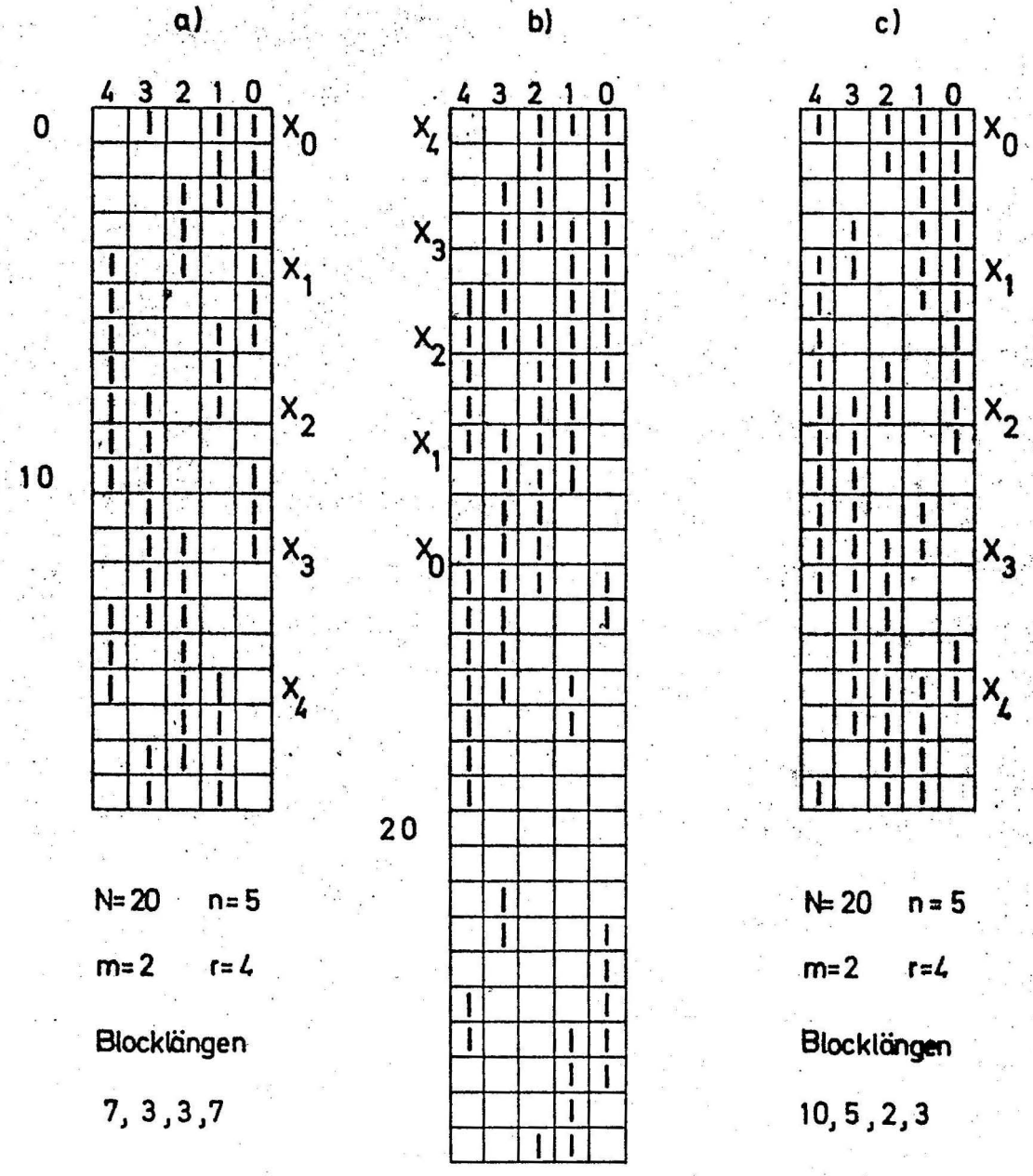


Abb.: 2 Drei Beispiele für einschrittige Kettencodes a) und b) nach TOOTILL

endliche Stellenzahl, so ist der Umfang des Zyklus ebenfalls auf diese oder eine kleinere Stellenzahl beschränkt (Abb.3).

n=5 (Stellenzahl)

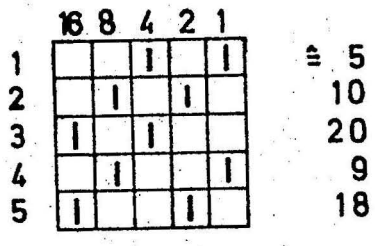


Abb.: 3 Umfang eines Zyklus

Als Repräsentanten des Zyklus nimmt man die kleinste auftretende natürliche Zahl; sie ist stets ungerade. Die Zyklen ordnet man nach den Gewichten w ihrer erzeugenden Binärfolgen. Am Beispiel der Stellenzahl 5 sei dieses gezeigt.

w=0

Es existiert nur ein Zyklus, der nur den einzigen Wert null enthält. Einen Zyklus, dessen Umfang kleiner ist als n, bezeichnen wir als "singulären" Zyklus.

w=1

Es existiert ebenfalls nur ein Zyklus; dieser nimmt aber alle Werte der 2-er Potenzen bis 2^{n-1} an.

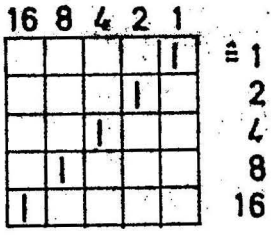


Abb.: 4 Der Zyklus „1"

Für die weiteren Gewichte ergibt sich durch einfache Überlegung folgende Formel für die Anzahl Z_w der möglichen Zyklen vom Gewicht w. Zunächst ist

$$Z_w = \frac{1}{n} \binom{n}{w}$$

Z_w muß immer eine ganze Zahl sein. Erhält man einen Rest, so liegen neben bestehenden vollständigen (regulären) Zyklen ein oder mehrere verkürzte (singuläre) Zyklen vor. Diese verkürzten Zyklen treten auf, wenn Stellenzahl n und Gewicht w des Zyklus einen oder mehrere gemeinsame Teiler haben.

Ein Beispiel dafür ist:

n=6 w=2 gemeinsamer Teiler t=2

$$Z_w = \frac{1}{6} \times \binom{6}{2} = \frac{1}{6} \times 15 = 2, \text{Rest } 3$$

Es existieren zwei reguläre Zyklen und ein verkürzter Zyklus.

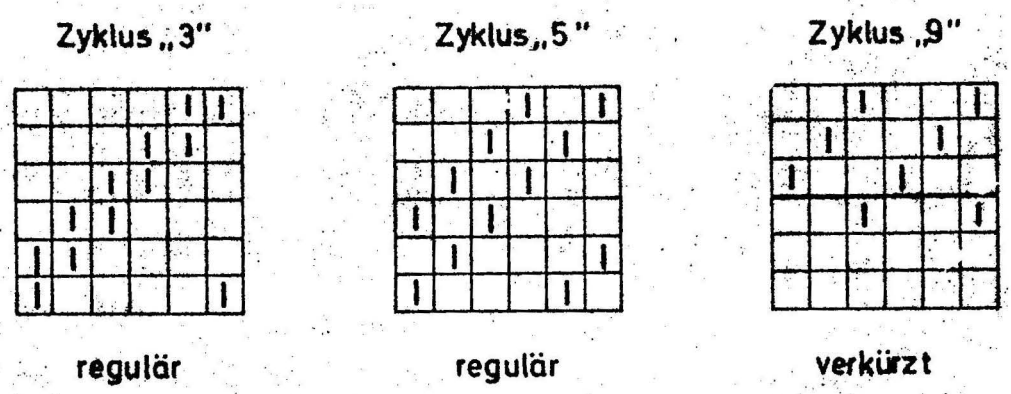


Abb.: 5 Beispiele für reguläre und verkürzte Zyklen

Für die Stellenzahl 5 erhält man für die einzelnen Gewichte folgende Anzahlen von Zyklen.

W	0	1	2	3	4	5
Z_W	1	1	2	2	1	1

0	1	2	3	4	5	Gewicht
		(3)	(7)			
[0]	(1)			(15)	[31]	
		(5)	(11)			

Tab.: 1 Zyklen für n=5

Hierbei stellen die im Kreis ◯ stehenden Zahlen die Repräsentanten der regulären Zyklen dar. Die Quadrate kennzeichnen singuläre, verkürzte Zyklen.

Verbindungen zwischen den Zyklen verschiedener Gewichte bestehen durch sogenannte Nachbarschaften. Entfernt man aus einem Repräsentanten eines Zyklus mit dem Gewicht w eine "Eins", so entsteht auch ein Wort eines Zyklus mit dem Gewicht w-1. Da man dieses w-mal durchführen kann, hat jeder Zyklus w sogenannte "untere" Nachbarn, von denen unter Umständen einige gleich sein können. Entfernt man z.B. aus dem Zyklus 7 mit dem Gewicht 3 die Eins mit der Wertigkeit 2^2, so entsteht der Repräsentant des Zyklus 3. Entfernt man dagegen die Eins mit der Wertigkeit 2^0, so entsteht auch ein Wert des Zyklus 3. Verschiebt man diesen Wert um eine Stelle nach rechts, so erhält man den Repräsentanten. Der dritte untere Nachbar ist der Repräsentant des Zyklus 5.

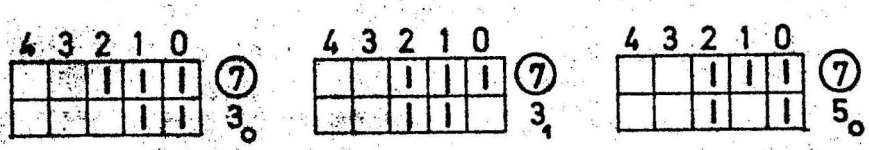


Abb.: 6 Beispiele für Nachbarschaften

Diese Nachbarschaft faßt man in einer Tabelle oder Grafik zusammen. In der nachfolgenden Grafik bedeutet jeder Verbindungsstrich eine Nachbarschaft. Die danebenstehende Zahl kennzeichnet die zum Erreichen des Repräsentanten notwendige Zahl der Verschiebungen.

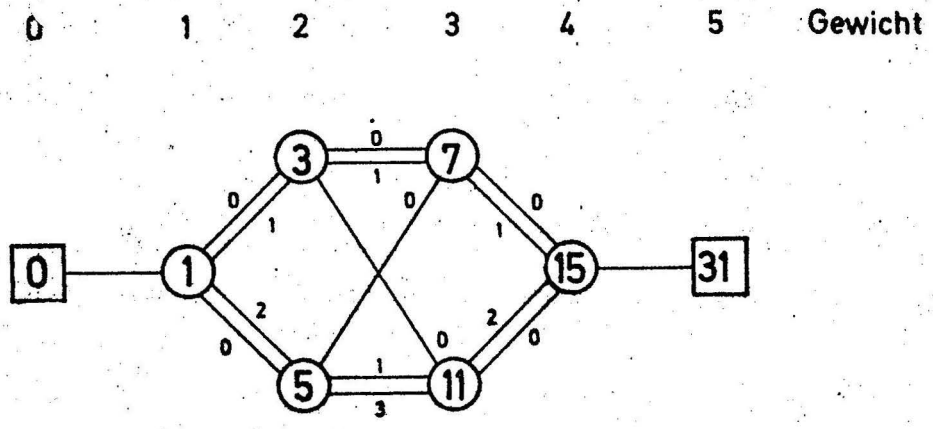


Abb.: 7 Verband der Zyklen für n=5

Einen einschrittigen Kettencode erhält man aus dieser Grafik, wenn man sich einen geschlossenen Weg über Nachbarschaften sucht, der der Bedingung genügt, daß die Summe der Verschiebungen auf diesem Weg +1 oder -1(mod 5) beträgt. Allgemein kann man schreiben

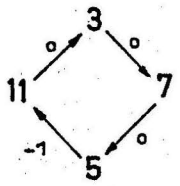
$$S = \pm 1 (\text{mod } n)$$

Dabei muß folgendes beachtet werden: bewegt man sich innerhalb des Verbandes der Zyklen in steigender Richtung, d.h. die Gewichte der Zyklen nehmen zu, so ordnet man diesen Verschiebungen ein negatives Vorzeichen zu, für die fallende Richtung ein positives Vorzeichen. Dadurch wird berücksichtigt, daß die Nachbarschaften zwischen den Zyklen nur als untere Nachbarschaften berechnet wurden.

Die Begründung für die oben angegebene Formel $S = \pm 1 \ (\text{mod } n)$ ist leicht einsehbar. Wie schon oben erwähnt, taucht ein Codewort nach r Wörtern im Codealphabet erneut, aber als zyklische Permutation auf. Es ist dabei entweder um eine Stelle nach rechts oder nach links verschoben.

Auch die Tatsache, daß bei dieser Art von Codekonstruktion ein einschrittiger Code entsteht, ist sofort zu sehen, da Zyklen, die einander benachbart sind, sich stets nur in einer Binärstelle unterscheiden können. Zum besseren Verständnis seien hier noch zwei Beispiele gezeigt:

a) verwendete Zyklen



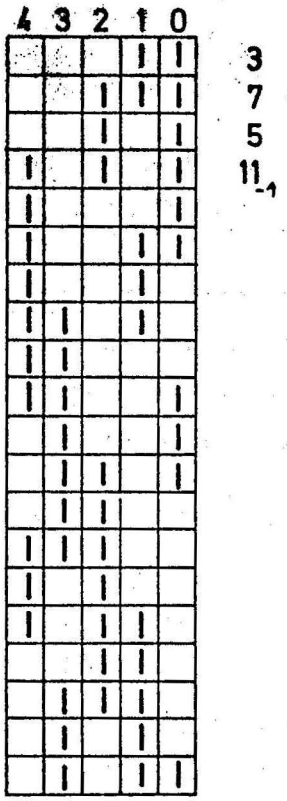
S = -1

Daten des entstehenden Kettencodes

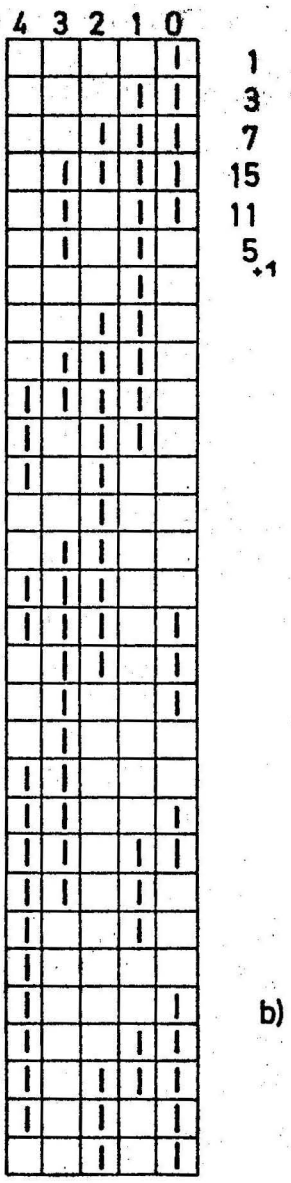
N=20 n=5 m=2 r=4

Blocklängen 7,3,3,7

Dieser Code entspricht dem von TOOTILL angegebenen Code.



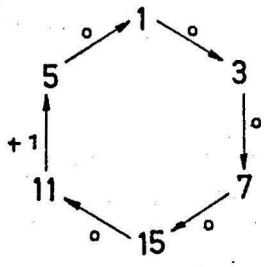
a)



b)

Abb.: 8 Zwei Beispiele einschrittiger Kettencodes aus Zyklen

b) verwendete Zyklen



S = +1

Daten des entstehenden Kettencodes

N=30 n=5 r=6 m=3

Blocklängen 10,3,2,2,3,10

Dieser Code erreicht fast den Maximalumfang $2^n = 32$.

1.4.1 Einschrittige Kettencodes mit d ≥ 3

Wie schon früher erläutert, bezieht sich d ≥ 3 auf Codewörter, die im Codealphabet um mindestens drei Schritte voneinander entfernt sind. Näher zusammenstehende Codewörter können aufgrund der Einschrittigkeit keinen höheren Abstand haben.

Um diese höheren Abstände zu erhalten, müssen die Zyklen selbst einen "inneren" Abstand von mindestens 3 haben. Aus diesem Grund kommen also nur Zyklen mit einem Gewicht w ≥ 3 infrage. Außerdem darf ein Zyklus nicht einen anderen mehrfach als Nachbarn haben oder einem singulären Zyklus benachbart sein, damit der innere Abstand d ≥ 3 gewährleistet ist. Zyklen, die diese Bedingungen erfüllen, haben einen inneren Abstand von mindestens 4 .

Für die Erzeugung eines Codes ist aber auch der Abstand der Zyklen untereinander wichtig. So haben z.B. zwei Zyklen mit w ≥ 3, die keinen gemeinsamen Nachbarn haben, mindestens den Abstand 4. Haben zwei Zyklen gleichen Gewichts einen gemeinsamen unteren Nachbarn, so haben sie mit Ausnahme einer bestimmten Stellung zueinander immer mindestens den Abstand 4. Der einmal auftretende

Abstand 2 ist Vorbedingung für die Einschrittigkeit des Codes (Beispiel a).

Haben dagegen zwei Zyklen gleichen Gewichts zwei gemeinsame untere Nachbarn, so tritt der Abstand 2 zweimal auf (Beispiel b).

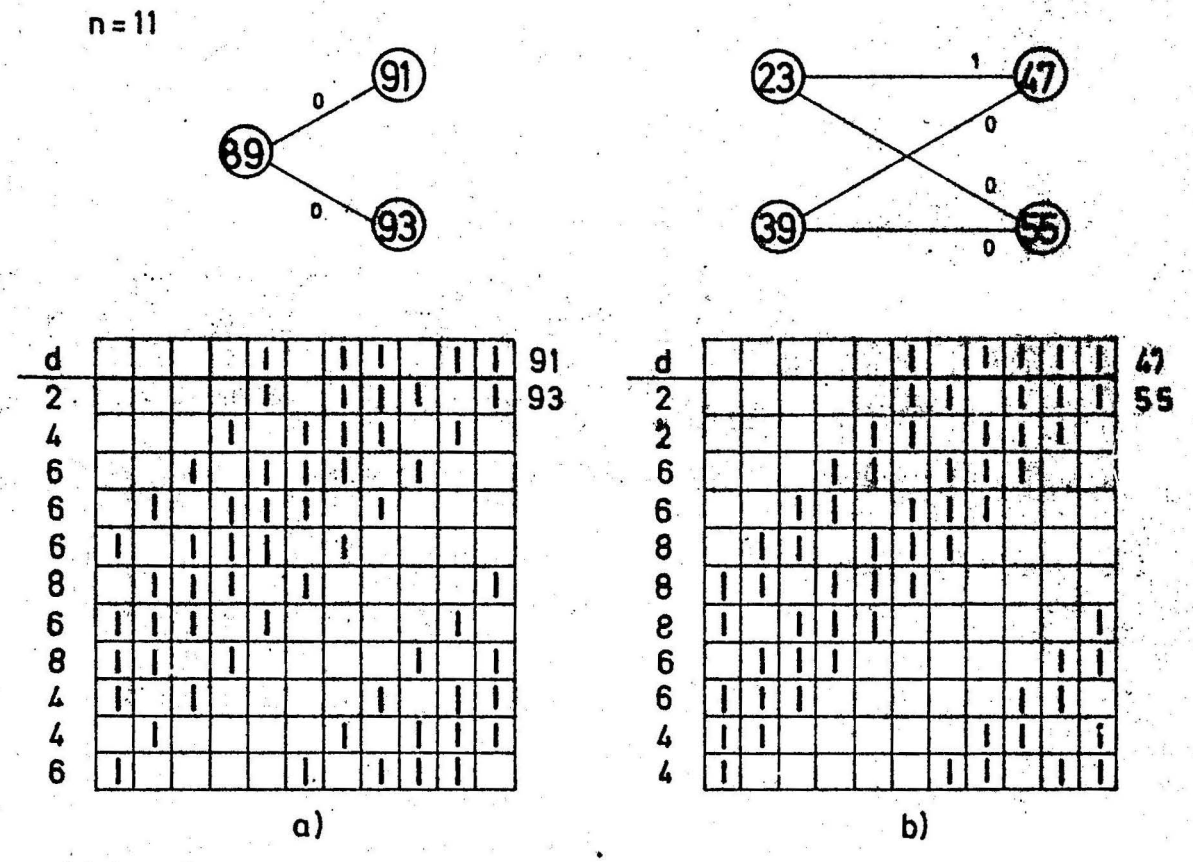
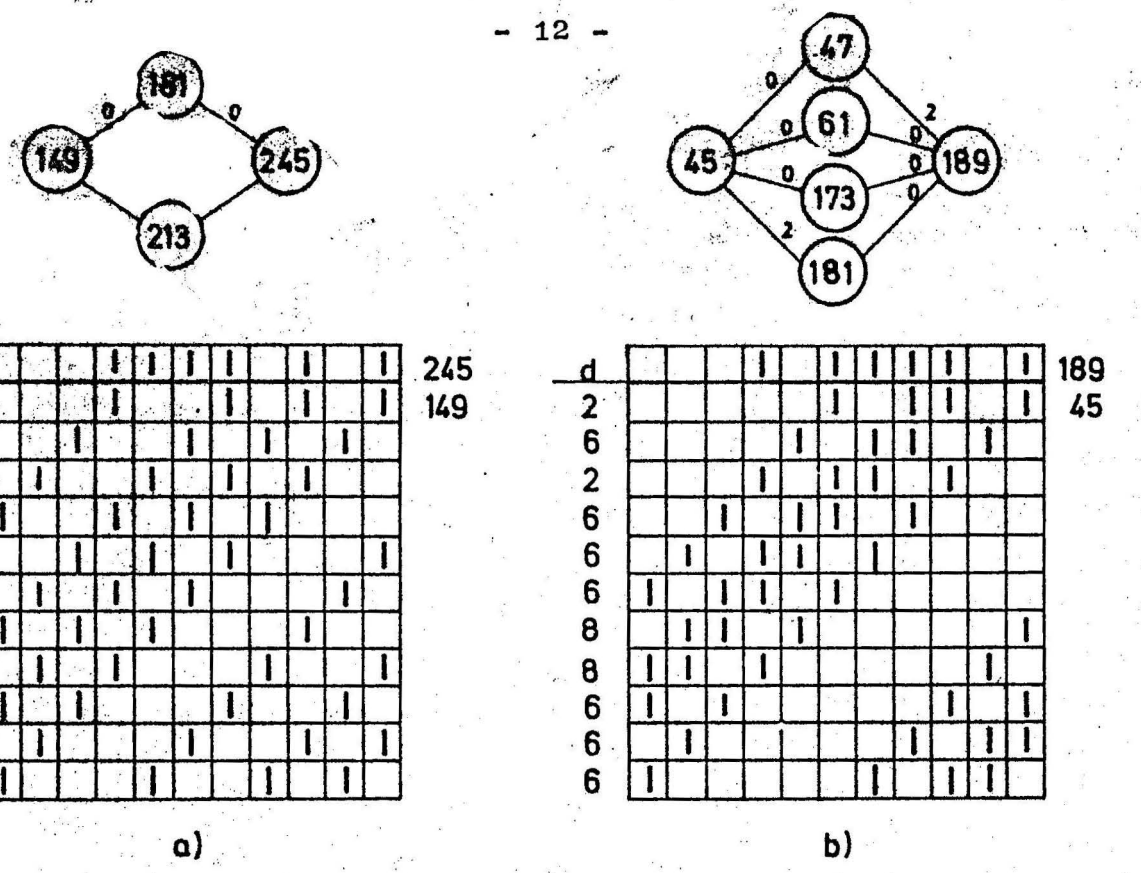


Abb.: 9 Beispiele für Nachbarschaften

Solche Zyklen erfüllen die geforderten Bedingungen nicht.

Für Zyklen, die sich im Gewicht um 2 unterscheiden, gibt es ebenfalls eine Verwendbarkeitsbedingung. Um den Minimalabstand $d \geq 3$ zu wahren, darf es nur zwei Wege geben, auf denen man die Zyklen miteinander verknüpfen kann (Beispiel a). Gibt es mehr als zwei Wege (Beispiel b), so wird der Minimalabstand nicht mehr eingehalten.



a) b)

Abb.: 10 Beispiele für Nachbarschaften

Berücksichtigt man bei der Auswahl der Zyklen alle diese Bedingungen, so sind einschrittige Kettencodes mit einem Minimalabstand von $d \geq 3$ zwischen Codewörtern, deren Abstand im Codealphabet ≥ 3 ist, konstruierbar. Solche Codes können zur Fehlererkennung und Fehlerkorrektur von Einzelfehlern verwendet werden. Folgende Einschränkung muß dabei aber gemacht werden. Da der Abstand zwischen aufeinanderfolgenden Codewörtern 1 ist, werden Einzelfehler, die ein Codewort erzeugen, nicht erkannt. In der Praxis bedeutet das, daß eine übertragene Amplitudenstufe um einen Quantisierungsschritt nach oben oder unten verfälscht werden kann. Dieser Fehler kann aber bei Übertragung von Sprache ohne weiteres zugelassen werden.

Einzelfehler, die nicht zu einem Codewort führen, können erkannt und korrigiert werden; die Korrektur ist aber unter Umständen zweideutig und kann dann auch einen um ± 2 verfälschten Wert ergeben.

1.4.2 Abschätzung des Maximalumfangs eines Codes für verschiedene Wortlängen

Bei der Berechnung der Anzahlen der Zyklen für verschiedene Wortlängen erhält man die nachfolgende Tabelle 2. Dabei gibt jeweils die erste Zahl die Anzahl der regulären Zyklen, die zweite die verkürzten Zyklen an. Der umrandete Teil kennzeichnet den Bereich der Zyklen, der für die Codekonstruktion aufgrund des Abstandes von Nullwort und Einswort infrage kommt. Die verkürzten Zyklen müssen dabei noch ausgeklammert werden. Der oben angegebene Bereich enthält ferner noch die Zyklen, die anderen doppelt benachbart sind. Die Anzahl dieser Zyklen entspricht der mit dem Gewicht 2, da diese auf jeden Fall mit dem Zyklus 1 doppelt benachbart sind und sich Doppelnachbarschaften durch den gesamten Verband der Zyklen fortpflanzen. Damit ergibt sich folgende Tabelle 3 für die Anzahl der verwendbaren Zyklen.

Eine grobe Abschätzung des möglichen Maximalumfanges eines Codes kann man durch folgende Überlegung erreichen. Jeder verwendete Zyklus der Stellenzahl n hat auch n Nachbarn (w untere Nachbarn und n-w obere Nachbarn). Zwei davon werden für den Code benutzt, die restlichen n-2 werden unbrauchbar. Da die verkürzten Zyklen bzw. Zyklen mit Doppelnachbarschaft und Zyklen der Gewichte 2 und n-2 auch Nachbarn der oben angegebenen Zyklen sein können, kann der Maximalumfang des Codes etwas größer als bei dieser Abschätzung werden. Die Tabelle 4 zeigt, daß für Wortlängen bis 10 keine Codes mit nennenswertem Umfang N zu erwarten sind. Die weiteren Untersuchungen beschränken sich deshalb auf n=11 .

w \ n	5	6	7	8	9	10	11
0	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
2	2	2+1	3	3+1	4	4+1	5
3	2	3+1	5	7	9+1	12	15
4	1	2+1	5	8+2	14	20+2	30
5	1	1	3	7	14	25+1	42
6		1	1	3+1	9+1	20+2	42
7			1	1	4	12	30
8				1	1	4+1	15
9					1	1	5
10						1	1
11							1

Tab.: 2 Anzahl der Zyklen

w \ n	7	8	9	10	11
3	2	3	5	7	10
4	2	4	10	15	25
5		3	10	20	37
6			5	15	37
7				7	25
8					10
Σ	4	10	30	64	144

Tab.: 3 Anzahl der verwendbaren Zyklen

n	a	b	c	S	N
9	30	4	28	32	36
10	64	4	32	36	40
		6	48	54	60
		8	64	72	80
11	144	4	36	40	44
		6	54	60	66
		8	72	80	88
		10	90	100	110
		12	108	120	132
		14	126	140	154
		16	144	160	176

a= vorhandene Zyklen
b= verwendete "
c = durch b unbrauchbar
S = b + c

Tab.: 4 Abschätzung des Codeumfangs

1.5 Erzeugung einer Familie von einschrittigen Kettencodes mit d ≥ 3 für n=11

Zur Lösung dieser Aufgabe kann man wie folgt vorgehen. Man erstellt sich eine Tabelle der Zyklen und ihrer Repräsentanten. Dazu dient das FORTRAN-Programm 1. Es erzeugt zuerst alle ungeraden Zahlen zwischen 1 und 2^{11} und berechnet die Gewichte der dualen Darstellung dieser Zahlen. Dabei werden nur Zahlen mit Gewichten zwischen 3 und 8 berücksichtigt. Anschließend werden die zyklischen Permutationen der berechneten Dualzahlen erzeugt und gespeichert. Danach folgt die Feststellung des Repräsentanten, d.h. der kleinsten Zahl innerhalb des Zyklus.

Das Unterprogramm SUCH 2 ordnet den Zyklus seinem Gewicht entsprechend ein und stellt dabei fest, ob er schon vorhanden ist.

Das Unterprogramm DRUCK 2 dient zusammen mit UEBER 2 zum Ausdrucken der berechneten Zyklen.

Mit dem FORTRAN-Programm 2 werden die Nachbarschaften zwischen den Zyklen bestimmt. Zuerst wird in diesem Programm festgestellt, an welchen Stellen der Repräsentanten sich mit "1" belegte Binärstellen befinden. Diese Stellen werden nacheinander gelöscht; dabei werden die unteren Nachbarn und eventuelle Verschiebungen berechnet. Die Ergebnisse werden gespeichert und am Ende des Programms ausgedruckt.

Ein drittes Programm dient zur Erzeugung der Codes. Da dieses Programm sehr umfangreich ist, soll hier nur kurz seine Arbeitsweise ohne Angabe der Befehlsliste geschildert werden.

Da ein Code mit einem bestimmten Umfang gesucht wird, muß man die Anzahl der zu verwendenden Zyklen festlegen.

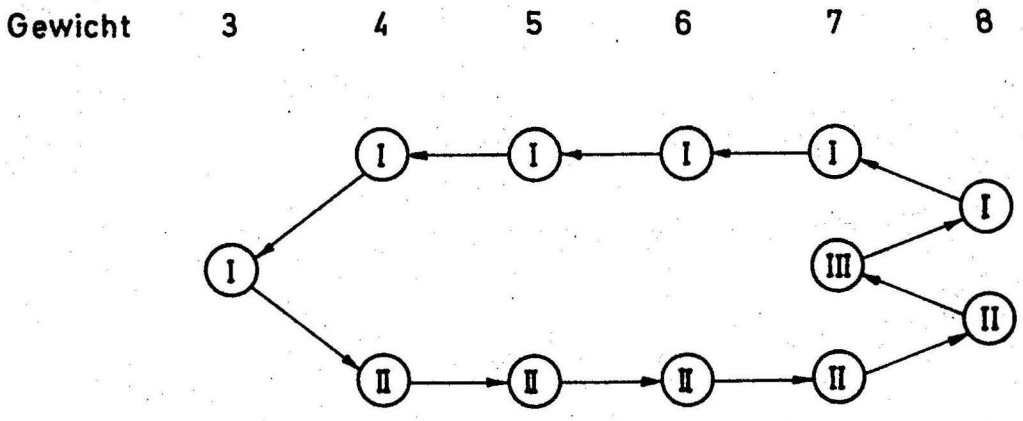


Abb.: 11 Beispiel für ein Suchschema

Außerdem wird ein Suchschema bestimmt. Hierbei kann jeder Zyklus der Startpunkt sein, z.B. auch der mit 8_I bezeichnete Zyklus. Anhand der Nachbarschaftstabelle sucht man sich nun einen Zyklus (7_I), der unterer Nachbar von 8_I ist. Im nächsten Schritt sucht man einen Zyklus 6_I, der unterer Nachbar von 7_I ist und den mit 8_I nur zwei mögliche Wege verbinden (siehe 1.4.1).

In weiteren Schritten werden dann die Zyklen niederer Gewichte bestimmt, wobei jeweils die Nachbarschaftsbedingungen zu beachten sind. So dürfen zum Beispiel die beiden verwendeten Zyklen vom Gewicht 5 (5_I und 5_{II}) keinen gemeinsamen Nachbarn haben.

Im letzten Teil des Programms wird dann ein Zyklus 7_{III} bestimmt, der neben den Nachbarschaftsbedingungen auch die in 1.4 gestellte Bedingung bezüglich der Verschiebung erfüllen muß. Das Programm endet mit dem Ausdrucken der für einen Code verwendbaren Zyklen.

Bei der Untersuchung des Verbandes der Zyklen mit 11 Bit wurden folgende Ergebnisse erzielt. Der größte erreichte Umfang lag bei 14 Zyklen, das entspricht einem Codeumfang von 154 Wörtern. Das Suchschema dazu hat folgende Form.

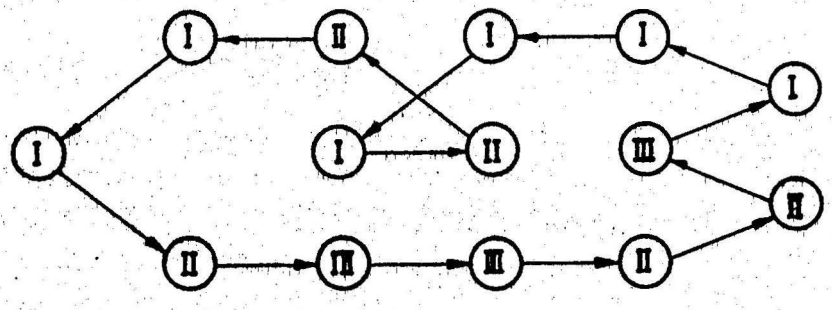


Abb.: 12 Suchschema für 14 Zyklen

Es wurden hierbei, ausgehend von 8_I, 28 mögliche, verschiedene Wege gefunden.

Der Codeumfang entspricht mit 154 Wörtern recht gut der in 1.4.2 gemachten Abschätzung.

In einem Aufsatz von SINGLETON /3/ wird als Maximalumfang für einen solchen Code bei einer Wortlänge von 11 Bit ein Wert von 96 Codewörtern angegeben.

Für das in Abbildung 11 angegebene Suchschema wurden, ausgehend von 8_I, 792 verschiedene Codes gefunden. Codes mit diesem Umfang (N=132) würden sich für die Übertragung eines Codealphabets von 128 Wörtern = 7 Bit gut eignen.

Für noch kleinere Umfänge, z.B. N=110, wächst die Anzahl der möglichen Wege im Verband der Zyklen rasch an. Die Auswertung lieferte für 10 Zyklen = N=110 etwa 10 000 mögliche Wege. Noch kleinere Umfänge wurden nicht untersucht, da sie technisch uninteressant erschienen.

Abschließend werden für verschiedene Umfänge einige Codes angegeben.

N	verwendete Zyklen
66	79, 207, 143, 175, 167, 335
88	39, 103, 119, 375, 187, 171, 163, 167
110	187, 191, 447, 415, 413, 103, 71, 69, 77, 93
132	37, 149, 213, 245, 491, 507, 251, 763, 351, 95, 47, 39
154	41, 169, 185, 187, 179, 243, 247, 759, 477, 509, 381, 95, 47, 45

Tab.: 5 Fünf Kettencodes

2. Praktische Anwendung eines einschrittigen Kettencodes

2.1 Verwendeter Code

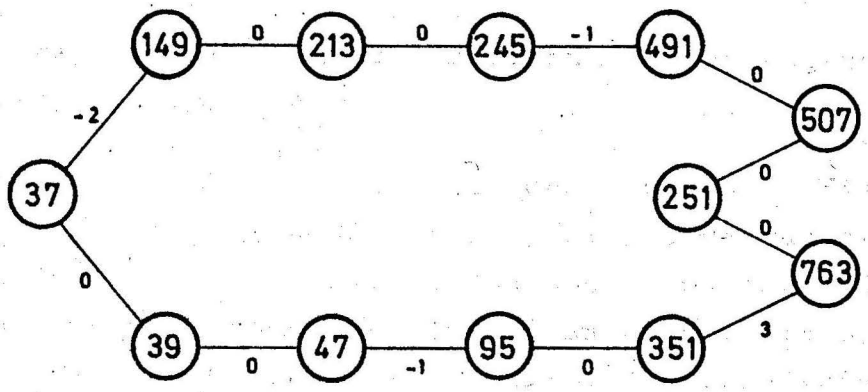
Wie schon in 1.4.1 erwähnt, kann man einen einschrittigen Kettencode als Kanalcode in einem PCM-System für Sprachübertragung benutzen.

In der vorliegenden Aufgabe wurde für die Quantisierung der Sprache ein 7-bit ADU mit 13-Segment-Kennlinie /4,5/ verwendet. Der Umfang des Kettencodes muß also $\geq 2^7 = 128$ sein. Infrage kommt dafür ein Code mit n=11 und 12 Zyklen $\hat{=}$ N = 132.

Abb. 13 zeigt die verwendeten Zyklen und die ersten Codewörter.

2.2 Zuordnung des Kettencodes

Für die Zuordnung eines 11 bit-Codewortes zu einem 7 bit-Wort des ADU gibt es mehrere Möglichkeiten. Man könnte zum Beispiel ein Abzählverfahren verwenden, welches nacheinander die Codewörter des Kettencodes erzeugt und bei Übereinstimmung des dualen Codewortzählers mit dem 7-bit-Wort des ADU die weitere Codeworterzeugung unterbricht. Dieses Verfahren hat den Nachteil , daß es entweder viel Zeit für die Umwandlung braucht oder mit sehr hoher Taktfrequenz arbeiten muß.



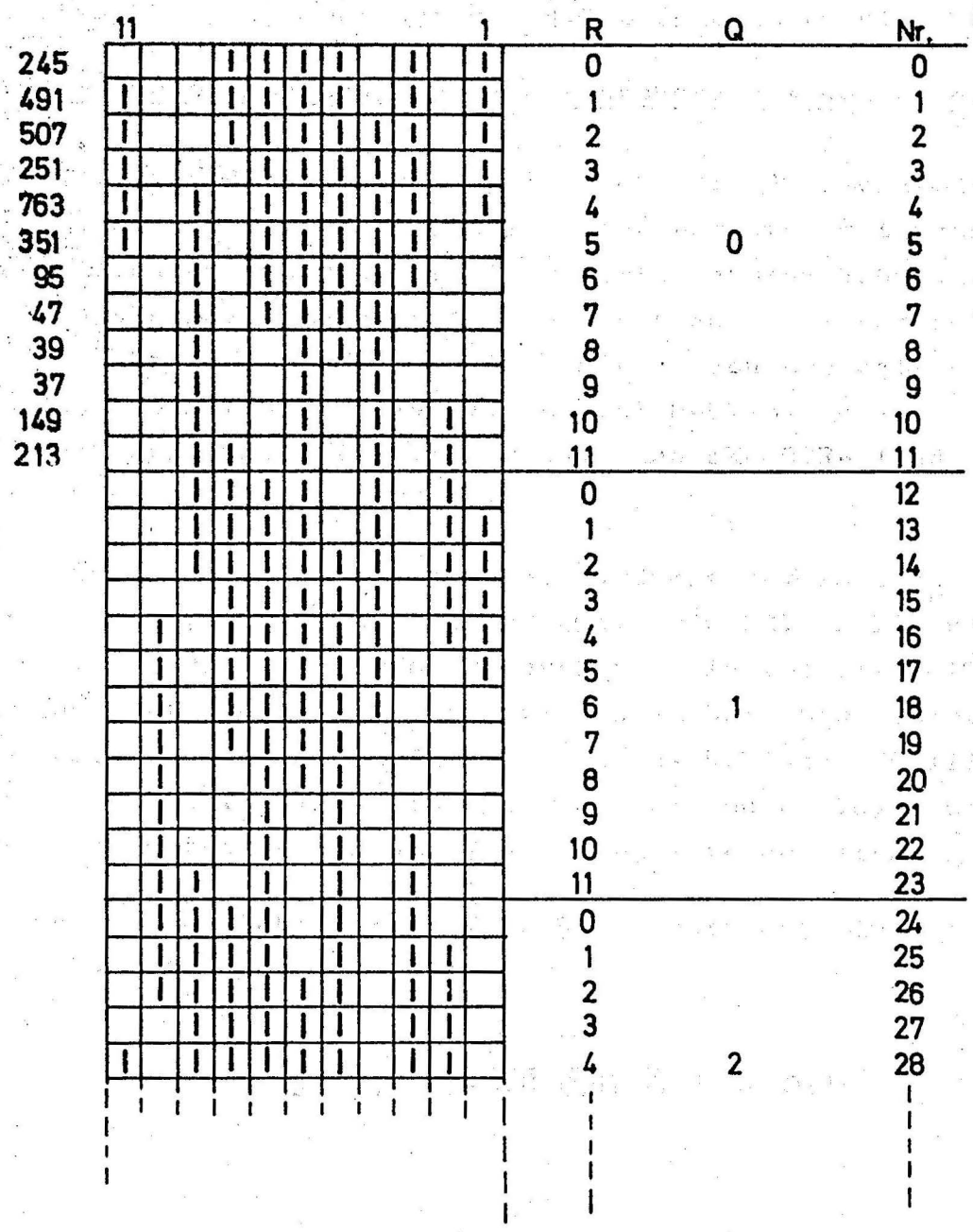


Abb.: 13 Einschrittiger Kettencode mit n=11 aus 12 Zyklen

In der vorliegenden Aufgabe wurde ein anderer Weg zur Lösung des Problems beschritten.

Man geht dabei von der Systematik des Codes aus. Wie schon früher erklärt, sind die Codewörter zyklische Permutationen der verwendeten Zyklen. Beim vorliegenden Code sind es 12 Zyklen, so daß jedes 13. Codewort demselben Zyklus angehört. Ein beliebiges Codewort kann also auch durch eine Zahl im Zwölfer-System lokalisiert werden, z.B. Codewort Nr. 27 27/12 = 2 Rest 3. Hierbei gibt der Rest den verwendeten Zyklus an, der Quotient kennzeichnet die Verschiebung gegenüber dem mit Q=0 bezeichneten Block (Abb. 13).

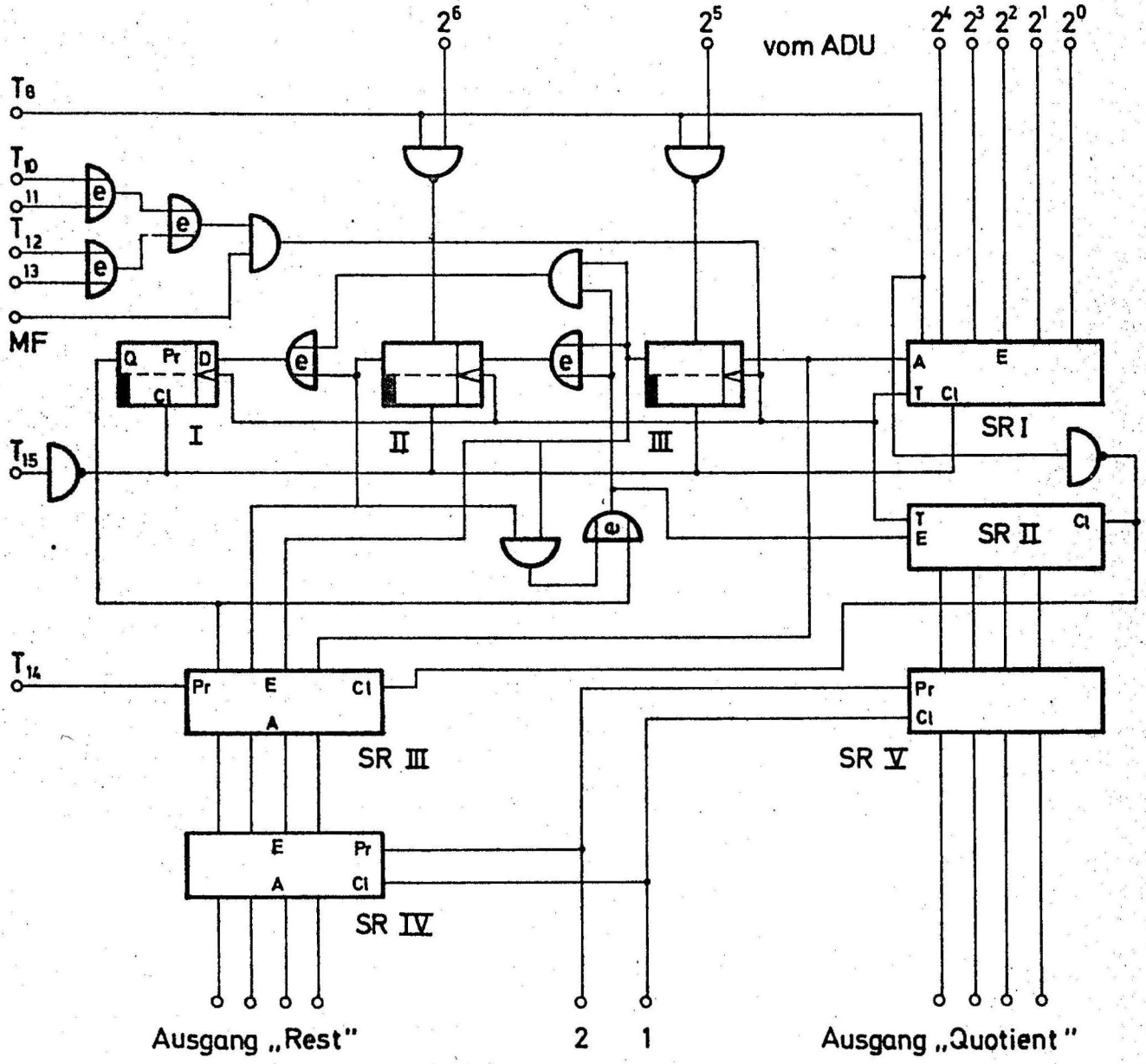
2.2.1 Dividierschaltung für die Zuordnung des Kettencodes

Abb. 14 zeigt den logischen Aufbau der Dividierschaltung. Die Division durch 12 wird von einem Schieberegister mit Rückkoppelungen vorgenommen und beruht auf einer fortgesetzten Subtraktion von 3, da in dualer Schreibweise 12 und 3 bis auf zwei Verschiebungen (2 mal Faktor 2) identisch sind. Dieses Schieberegister besteht aus einem 5-Bit-SR (SN 7476) und drei D-Flip-Flops ($1^1/2$ FJJ 131). Die Rückkoppelung erfolgt über UND- und EX-ODER-Glieder (SN 7408 bzw. SN 7486).

Zu Beginn des Divisionsvorganges wird mit dem Takt T_8 das vom ADU vorliegende 7-Bit-Wort stellenrichtig von SRI, D III und D II übernommen. Die Division erfolgt durch vier Taktschritte (T_{10}--T_{13}); der Quotient steht nach der Division im Schieberegister SR II, der Rest wird durch den Takt T_{14} in das Register SR III übernommen. Die Arbeitsweise der Schaltung soll durch logische Gleichungen, Wahrheitstabellen und ein Beispiel verdeutlicht werden.

Am Eingang des Schieberegisters SR II entsteht ein Signal nach folgender Gleichung:

$$ESR\,II = QI \cdot (\overline{QII \cdot QIII}) + \overline{QI} \cdot (QII \cdot QIII)$$



Pr = Preset
Cl = Clear
T = Takt
E = Eingang
A = Ausgang

Abb.: 14 Dividierschaltung

	QI	QII	QIII	ESR II
1	0	0	0	0
2	0	0	1	0
3	0	1	0	0
4	0	1	1	1
5	1	0	0	1
6	1	0	1	1
7	1	1	0	
8	1	1	1	

Tab.: 6
Wahrheitstabelle für ESR II

Die Zustände 7 und 8 treten nicht auf.

Für die D-Eingänge der drei D-Flip-Flops lauten die Eingangsbedingungen.

$$DIII = QSR\,I \qquad \text{(Ausgang des Schieberegisters I)}$$

$$DII = ESR\,I \cdot \overline{QIII} + \overline{ESR\,II} \cdot QIII$$

$$DI = QII \cdot (\overline{ESR\,II \cdot QIII}) + \overline{QII} \cdot (ESR\,II \cdot QIII)$$

Damit erhält man folgende Tabelle:

(t_n kennzeichnet den Zustand vor dem Taktimpuls

t_{n+1} " " " nach " ")

	QI	QII	QIII	QI	QII	QIII
1	0	0	0	0	0	A
2	0	0	1	0	1	"
3	0	1	0	1	0	"
4	0	1	1	0	0	"
5	1	0	0	0	1	"
6	1	0	1	1	0	"
7	1	1	0			
8	1	1	1			
	t_n			t_{n+1}		

Tab.: 7
Flip-Flop-Zustände vor und nach einem Taktimpuls

Auch hier treten die Zustände 7 und 8 nicht auf, da QI = QII = 1 nach einem Taktimpuls nicht möglich ist.

In Tab. 6 kann man sehen, daß ESR II immer dann "1" ist, wenn die durch QI, QII und QIII dargestellte Dualzahl ≥ 3 ist. Dann ist die oben erwähnte Subtraktion von 3 ausführbar, und die entsprechende Stelle im Quotienten wird "1" .

Die Durchführung der Subtraktion wird in Tab. 7 deutlich. Sie wird ausgeführt, wenn die Dualzahl ≥ 3 ist; das Ergebnis der Subtraktion erscheint nach dem Takt (t_{n+1}) um eine Stelle nach links verschoben. Als Beispiel sei die Division von 115 durch 12 gezeigt.

115 : 12 = 9 Rest 7
108
7

	QI	QII	QIII	SR I					ESR II	SR II			
115	0	1	1	1	0	0	1	1	1	0	0	0	0
nach t_{10}	0	0	1	0	0	1	1	0	0	1	0	0	0
" t_{11}	0	1	0	0	1	1	0	0	0	0	1	0	0
" t_{12}	1	0	0	1	1	0	0	0	1	0	0	1	0
" t_{13}	0	1	1	1	0	0	0	0		1	0	0	1
	Rest ≙ 7									Quotient ≙ 9			

Tab.: 8 Beispiel einer Division

Das Ergebnis der Division (Quotient und Rest) wird durch die Befehle "Clear" und "Preset" an 1 und 2 in die Register SR IV und SR V übernommen.

2.3. Erzeugung der Codewörter

Das logische Schaltbild Abb. 15 zeigt die Schaltung zur Erzeugung der Codewörter.

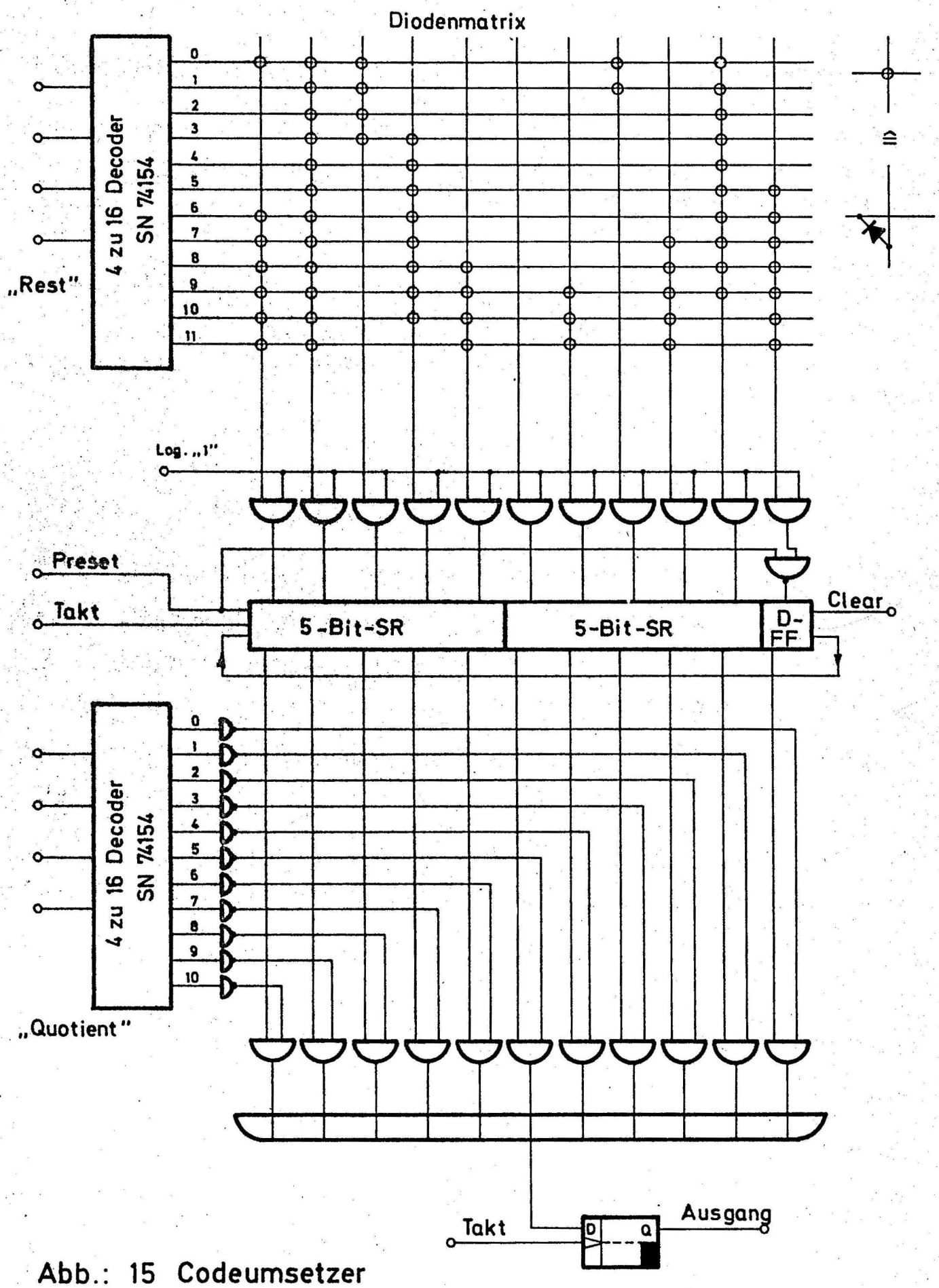


Abb.: 15 Codeumsetzer

Codebestimmendes Element ist die Diodenmatrix, deren 12 Zeilen den ersten 12 Codewörtern entsprechen. Der 4 zu 16 Decoder (SN 74154) für das Dualwort des Restes steuert die entsprechende Zeile der Matrix an. Über UND-Glieder wird das in dieser Zeile gespeicherte Codewort in ein 11-Bit Schieberregister (2 x SN 7496 und $^1/_2$ F JJ 131) übernommen. Da Serienausgang und Serieneingang dieses Schieberegisters miteinander verbunden sind, kann durch Anlegen eines Taktes der Inhalt des Registers zyklisch verschoben werden. Ein zweiter 4 zu 16 Decoder für den Quotienten bereitet eines der elf UND-Glieder vor. Über dieses UND-Glied gelangt das Codewort mit der entsprechenden Verschiebung (Abb. 16) über das ODER-Glied und das D-Flip-Flop an den Ausgang. Dort kann es zur weiteren Verarbeitung in serieller Form entnommen werden.

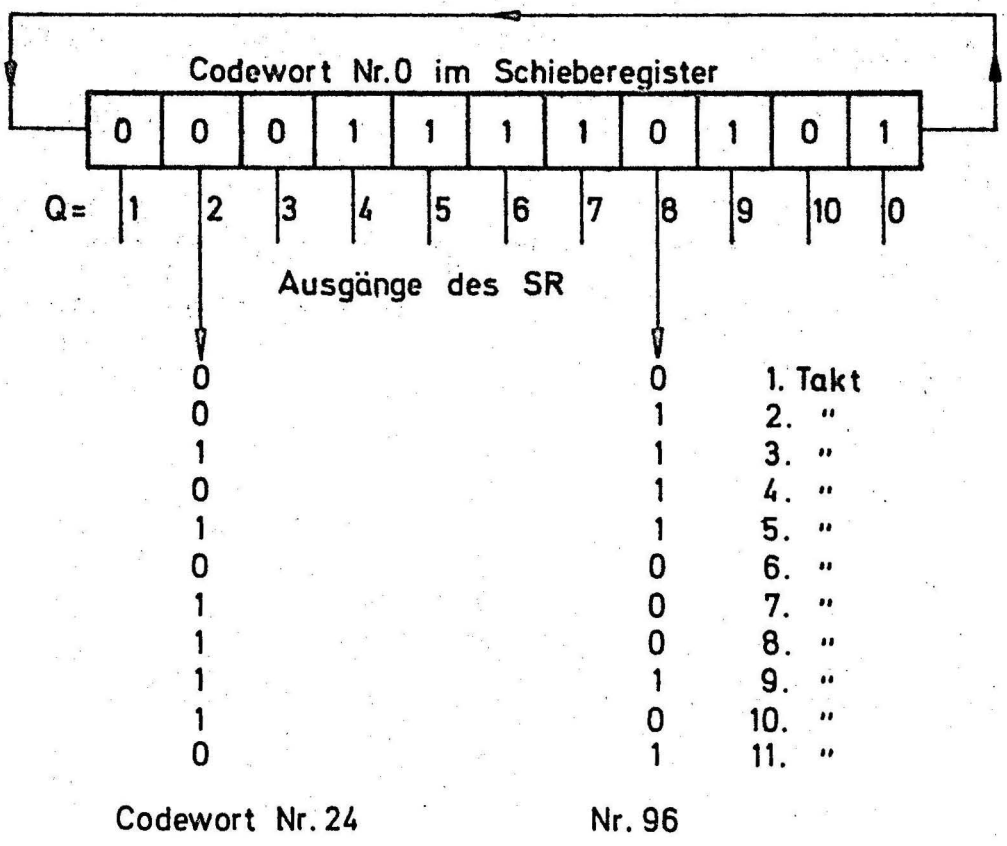


Abb.: 16 Erzeugung der Verschiebung

```
C       PROGRAMM 1
C       BERECHNUNG DER REPRAESENTANTEN UND ZYKLEN FUER 11 BIT
        DIMENSION NW(11),N(11),N3(165),N4(330),N5(642),N6(462)
        DIMENSION N7(330),N8(165)
        DATA IA,IB,IC,ID,IE,IFF/6*0/
        IWERT1 = 1
   10   IY = 1
        IWERT = IWERT1 + 2
        IF (IWERT .GT. 2048) GO TO 190
        IWERT1 = IWERT
        IG = 0
        IX = 0
   30   IWERT = IWERT * 2
        IX = IX + 1
        IF (IX .EQ. 12) GO TO 60
        IF (IWERT .LE. 2047) GO TO 30
        IG = IG + 1
        IWERT = IWERT - 2048
        GO TO 30
   60   IF (IG .LT. 3 .OR. IG .GT. 8) GO TO 10
        IX = 1
        IWERT2 = IWERT1
   80   NW(IX) = IWERT2
        IF (IX .EQ. 11) GO TO 90
        IWERT2 = IWERT2 * 2
        IF (IWERT2 .LT. 2048) GO TO 100
        IWERT2 = IWERT2 - 2047
  100   IX = IX + 1
        GO TO 80
   90   IX = 1
        IR = 1
        IWERT = NW(IX)
        IX = 2
  110   IF (IX .EQ. 12) GO TO 130
        IF (NW(IX) .GE. IWERT) GO TO 120
        IWERT = NW(IX)
        IR = IX
  120   IX = IX + 1
        GO TO 110
  130   N(IY) = IWERT
  140   IY = IY + 1
        IF (IR .EQ. 11) IR = 0
        IR = IR + 1
        N(IY) = NW(IR)
        IF (IY .NE. 11) GO TO 140
        IF (IG .EQ. 3) CALL SUCH2 (IA,N,N3,165,&10)
        IF (IG .EQ. 4) CALL SUCH2 (IB,N,N4,330,&10)
        IF (IG .EQ. 5) CALL SUCH2 (IC,N,N5,462,&10)
        IF (IG .EQ. 6) CALL SUCH2 (ID,N,N6,462,&10)
        IF (IG .EQ. 7) CALL SUCH2 (IE,N,N7,330,&10)
        IF (IG .EQ. 8) CALL SUCH2 (IFF,N,N8,165,&10)
```

```
190 CALL DRUCK2 (N3,165,3)
    CALL DRUCK2 (N4,330,4)
    CALL DRUCK2 (N5,462,5)
    CALL DRUCK2 (N6,462,6)
    CALL DRUCK2 (N7,330,7)
    CALL DRUCK2 (N8,165,8)
    CALL EXIT
    END



    SUBROUTINE SUCH2 (IA,N,NI,I,*)
    DIMENSION NI(I),N (11)
    IY = 1
    F = N(IY)
    JA = IA + 1
 10 JA = JA - 11
    IF (JA .LT. 1) GO TO 30
    IF (F .EQ. NI(JA)) RETURN 1
    IF (JA .NE. 1) GO TO 10
 30 IA = IA + 1
    NI(IA) = N(IY)
    IF (IY .EQ. 11) RETURN 1
    IY = IY + 1
    GO TO 30
    END



    SUBROUTINE DRUCK2 (NI,I,J)
    DIMENSION NI(I)
    IL = I/11
    CALL UEBER (J)
    LL = 0
    DO 50 L=1,IL
    LL = LL + 1
    IF (LL .LE. 50) GO TO 20
    LL = LL - 50
    CALL UEBER (J)
 20 LA = 11 * (L - 1) + 1
    LE = LA + 10
 50 WRITE (8,10) L, (NI(K),K=LA,LE)
 10 FORMAT (4X,I3,5X,11I7)
    RETURN
    END
```

```
      SUBROUTINE UEBER (J)
      WRITE (8,10) J
   10 FORMAT (1H1,4X,,GEWICHT =,,I3,//2X,, LFD. NR.,/)
      RETURN
      END
```

```
C     PROGRAMM 2
C     BERECHNUNG DER NACHBARSCHAFTEN FUER 11 BIT
      IMPLIZIT INTEGER (A-Z)
      DIMENSION Z(4000),A1(462),A2(462),ZE( 6),Y( 4000 ),V(4000),W(4000)
      DATA ZE/165,330,462,462,330,165/
      READ (3,500) (A1(J),J=1,165)
      A = 1
   10 IF (A-1) 20,15,20
   15 GA1 = 165
      GA2 = 330
      GO TO 70
   20 IF (A-2) 30,25,30
   25 GA1 = 330
      GA2 = 462
      GO TO 70
   30 IF (A-3) 40,35,40
   35 GA1 = 462
      GA2 = 462
      GO TO 70
   40 IF (A-4) 50,45,50
   45 GA1 = 462
      GA2 = 330
      GO TO 70
   50 GA1 = 330
      GA2 = 165
   70 NR = ZE(A+1)
      READ (3,500)  (A2(J),J=1,NR)
      U = 1
      Q = 1
  100 X = 11
      WERT = A2(Q)
      ZW1 = WERT
  110 WERT = WERT * 2
      X = X - 1
      IF (X) 130,120,120
  130 Q = Q + 11
      IF (Q - GA2) 100,100,300
  120 IF (WERT .LT. 2048) GO TO 110
      ZW1 = A2(Q) - 2**X
      P = 1
```

```
140 IF (ZW1 - A1(P)) 145,170,145
145 IF (P - GA1) 160,150,160
150 PRINT 501
    GO TO 99
160 P = P + 1
    GO TO 140
170 Y(U) = (Q-1) /11 + 1
    V(U) = 1 + P/11
    W(U) = MOD(P,11)
    VV = V(U) * 64
    YV = Y(U) * 32768
    Z(U) = W(U) + VV + YV
    U = U + 1
    WERT = WERT - 2048
    GO TO 110
300 U = U - 1
    SZ = A + 3
    WRITE (8,503)
    DO 400 J=1,U,SZ
    JJ = J + SZ - 1
400 WRITE (8,505)  (Y(I),V(I),W(I),I=J,JJ)
    WRITE (4,502) (Z(I),I=1,U)
    IF (A - 5) 310,99,310
310 A = A + 1
    DO 315 J=1,462
315 A1(J) = A2(J)
    GO TO 10
 99 CALL EXIT
500 FORMAT (1115)
501 FORMAT (,HIER STIMMT WAS NICHT,)
502 FORMAT (1018)
503 FORMAT (1H1)
505 FORMAT (10(3X,I3,1X,I3,1X,I2))
    END
```

Literaturverzeichnis

/1/ E.R. BERGER — Nachrichtentheorie und Codierung in Taschenbuch der Nachrichtenverarbeitung Springer-Verlag/Berlin-Heidelberg-New York

/2/ G.G. TOOTILL — The use of cyclic-permuted chain codes for digitisers Intern. Conf. Inf. Process. (ICIP) Paris 1959 S. 414 - 419

/3/ R.C. SINGLETON — Generalized snake-in-the-box codes IEEE Transactions on Electronic Computers Vol. EC 15 No. 4 1966 p. 596 - 602

/4/ L. SCHWEIZER — An Experimental Pulscode Modulation Encoder for 24 Speech Channals IEEE COM-17 No. 5 1969 p. 592 - 594

/5/ H. HESSENMÜLLER, H.W. WELLHAUSEN — Die Quantisierungsverzerrungen in PCM-Systemen bei Verwendung der 13-Segment-Kompressor-Kennlinie Technischer Bericht des FTZ FTZ A 442 TBr 9 1968

Mein Dank gilt

Herrn cand.-ing. J. Schlotter für seine wertvolle Hilfe bei der Programmierung und der Durchführung der Rechnung sowie der Deutschen Forschungsgemeinschaft, die durch Bereitstellung der Personal- und Sachmittel die Durchführung dieser Arbeit ermöglichte.