

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam,
The Netherlands.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications; it is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O) and the Central Organization for Applied Scientific Research in the Netherlands (T.N.O), by the Municipality of Amsterdam, by the University of Amsterdam, by the Free University at Amsterdam, and by industries.

MC SYLLABUS

7.1

MC SYLLABUS

7.1

COLLOQUIUM MODERNE REKENMACHINES

DEEL I

MATHEMATISCH CENTRUM AMSTERDAM

1969

In deze uitgave in de Syllabusreeks van het Mathematisch Centrum wordt de tekst weergegeven van voordrachten die in de jaren 1952 - 1959 in zaterdagochtendbijeenkomsten zijn gehouden binnen het kader van een colloquium Moderne Rekenmachines.

De tekst van deze voordrachten is overgenomen van de destijds uitgegeven gestencilde syllabussen, waarbij om het historische aspect te bewaren alleen duidelijk herkenbare typefouten zijn gecorrigeerd.

De reproductie heeft zich uiteraard moeten beperken tot die voordrachten waarvan een syllabus werd vervaardigd en nog een exemplaar beschikbaar was.

Desondanks is de omvang zo groot geworden dat de uitgave plaatsvindt in twee delen.

Door uitbreiding van de activiteiten, gecombineerd met internationale contacten op rekenmachinegebied in de vorm van IFIP is uit genoemd colloquium voortgekomen het Nederlands Rekenmachine Genootschap.

Ter gelegenheid van het 10-jarig bestaan van deze vereniging op 25 april 1969 werd in samenwerking met het Mathematisch Centrum een lustrumnummer uitgegeven, waarvan de inhoud overeenstemt met die van deze uitgave.

Inhoudsopgave Deel I

B.J. Loopstra	Logische synthese van rekencircuits	1
H.J.A. Duparc	Coderingsproblemen	10
B.J. Loopstra	Het gebruik van ferromagnetische materialen in rekencircuits	21
W.L. van der Poel	Welke opdrachten zijn essentieel in een rekenmachine?	27
S.J. Duinker	Dynamische magnetische flip-flops	34
C.S. Scholten	Een of meer adrescodes in rekenmachines?	41
G.A. Blaauw	De Mark IV rekenmachine van Harvard University	52
E.W. Dijkstra	Arithmetische methodes, ten dienste van de programmering	58
N.C. de Troye	Gelijkrichters	65
A. van Wijngaarden	Rekenmethodes voor automatische rekenmachines	73
H.A. Lauwerier	Het gebruik van de B-tube bij de Ferranti computers	81
W.L. van der Poel	De Ptera	88
B.J. Loopstra	Tellers	95
C.S. Scholten	Enige hulpmiddelen bij het oplossen van binaire vergelijkingen en toepassingen daarvan	100
E.W. Dijkstra	Controle op binaal-decimale conversie	111

A. van Wijngaarden		
	Dynamica van rekenmachines	120
H.J.A. Duparc		
	From recurring fraction to recurring computing circuit	129
G.A. Blaauw		
	De A.R.R.A.	138

LOGISCHE SYNTHESE VAN REKENCIRCUITS

B.J. Loopstra

Voordracht in het colloquium Moderne Rekenmachines van het Mathematisch Centrum op 19 april 1952 te Amsterdam.

Het probleem, hoe men met behulp van een beperkt aantal types schakelelementen zoals bijv. relais, pentodes en triodes, betrekkelijk gecompliceerde schakel- of arithmetische operaties kan verrichten is in het algemeen met enige scherpzinnigheid wel op te lossen. De vraag echter of met de gevonden oplossing ook de minimaaloplossing is gevonden, d.w.z. die oplossing, welke het kleinste aantal schakelelementen bevat, is bij de huidige stand van kennis op dit gebied, slechts zelden met zekerheid te beantwoorden.

Bij de verder te stellen vraag welke eventueel hypothetische schakelelementen zich het beste zouden lenen om bepaalde klassen van problemen op te lossen, tast men vrijwel geheel in het duister.

We zullen ons er dan ook toe bepalen na te gaan wat er met betrekking tot het vinden van minimaal oplossingen met gegeven elementen vooral door Aiken en zijn groep in Harvard is gedaan en aan de hand hiervan laten zien, welke desiderata er op dit gebied nog zoal zijn.

We zullen ons beperken tot binaire variabelen, d.w.z. variabelen welke slechts twee discrete waarden kunnen aannemen. Deze twee waarden zullen we voor elke variabele aangeven als 0 en 1. De variabelen zelf noemen we x , y , z ... en we zetten

$$x' = 1 - x.$$

Uit deze beperkingen volgt dat bij een probleem met n onafhankelijk veranderlijken, in den vervolge technisch aangeduid als "ingangen" er een eindig aantal, nl. 2^n verschillende ingangswaarden mogelijk zijn.

Evenzo zijn er bij m afhankelijke variabelen of uitgangen 2^m verschillende uitgangswaarden mogelijk. Hierbij dient echter wel te worden opgemerkt, dat dit aantal niet altijd bereikt hoeft te worden. Het is bijv. al dadelijk in te zien, dat het aantal verschillende uitgangswaarden nooit groter kan zijn dan het aantal verschillende ingangswaarden en deze situatie doet zich o.a. voor bij het later te bespreken probleem der decodatie, waar geldt dat $m = 2^n$. Men kan dus een schakeling, waarbij de variabelen aan deze beperkingen voldoen, opvatten als een operator, welke aan 2^m binaire getallen 2^n andere getallen toevoegt, waarvan er ten hoogste 2^m verschillend zijn.

Tenslotte rest ons nog beperkingen op te stellen ten aanzien van de te gebruiken schakelementen.

Voorlopig zullen we ons houden aan de volgende:

$$T_n(x_i) = \prod_{i=1}^n (1 - x_i)$$

$$C_n(x_i) = 1 - T_n(x_i)$$

$$P_2(x_1, x_2) = 1 - x_1 x_2.$$

De letters T, C en P staan hier respectievelijk voor triode, cathodevolger en pentode. Dit is echter voor de volgende theoretische beschouwingen van geen enkel belang.

In de eerste plaats zullen we nu beschouwen het geval $m = 1$, één uitgang aan het circuit dus.

In dit geval kan de werking van de schakeling dus worden gespecificeerd door aan te geven voor welke combinaties der ingangen de uitgang 1 zal zijn. We krijgen dan een lijstje, bijv. voor $n = 2$:

x_1	x_2	Uitgang
0	0	$0 = f_0$
0	1	$1 = f_1$
1	0	$1 = f_2$
1	1	$0 = f_3$

Uit dit lijstje is gemakkelijk een formule voor de uitgang $f(x_1, x_2)$ te vinden. Blijkbaar geldt immers:

$$f(x_1, x_2) = x_1' x_2' f_0 + x_1' x_2 f_1 + x_1 x_2' f_2 + x_1 x_2 f_3$$

in ons geval dus $x_1' x_2 + x_1 x_2'$.

Stelt men algemeen:

$$p_0^n = x_1' x_2' \dots x_n'$$

$$p_1^n = x_1' x_2' \dots x_n$$

$$\vdots$$

$$p_{v-1}^n = x_1 x_2 \dots x_n \quad (v = 2^n).$$

Dan is $f(x) = \sum_{i=0}^{v-1} p_i^n f_i$ de canonische vorm, waarop iedere functie van n ingangsvariabelen kan worden gebracht.

Het is duidelijk dat de $f_0 \dots f_3$ van ons voorbeeld op 16 verschillende wijzen kunnen worden gekozen en in het algemeen geldt, dat voor n ingangsvariabelen dit aantal "schakelfuncties" $N = 2^{2^n} = 2^v$ bedraagt.

Wij hoeven echter niet al deze N schakelfuncties in onze beschouwingen te betrekken.

Permutatie der ingangsvariabelen is technisch gezien gewoonlijk een kwestie van het verwisselen van draadjes en de aanname, dat x_i en x_i' beide beschikbaar zijn is zeer vaak geoorloofd en zullen we daarom maken, zodat we bij een functie reeds $n!2^n$ mogelijkheden hebben om de variabelen te ordenen. Nu zullen deze over het algemeen geen aanleiding geven tot $n!2^n$ verschillende uitgangsfuncties in verband met b.v. symmetrie-eigenschappen van de oorspronkelijke functie, maar we vinden dus uit de ene functie een aantal schakelfuncties van de 2^v , welke er zijn. Van belang is natuurlijk in hoofdzaak de verzameling der σ_n functies, waaruit door het toepassen van de $n!2^n$ transformaties alle 2^v mogelijke functies zijn af te leiden. Een betere methode om deze σ_n functies te vinden dan ze te laten uitzoeken door een rekenmachine en een betere

weg om het getal σ_n te bepalen dan het afwachten van het resultaat der machine is op het ogenblik helaas niet beschikbaar. Bekend is tot dusverre slechts $\sigma_2 = 3$, $\sigma_3 = 22$, $\sigma_4 = 402$.

Dat deze σ_n functies in deze zin onafhankelijk zijn, wil nog niet zeggen, dat ook de fysische circuits waarmee deze functies te realiseren zijn, alle verschillend moeten zijn.

Voorbeeld: (voor $n = 3$)

$$x'yz + xy'z' + xyz \quad \text{en} \quad x'yz + xy'z + xyz' + xyz$$

welke resp. geschreven kunnen worden als

$$P_2(y', z) \cdot P_2(y, z') \cdot P_2(x', y')$$

en

$$P_2(y', z') \cdot P_2(x', y') \cdot P_2(x', z').$$

Beide functies kunnen dus met dezelfde schakeling van 3 pentodes gerealiseerd worden.

We zullen nu na moeten gaan hoe we met binaire functies in het algemeen kunnen rekenen om te vinden hoe gegeven functies uit binaire standaardfuncties zouden kunnen worden opgebouwd.

Hierbij moeten we bedenken, dat we met binaire grootheden alleen die manipulaties mogen uitvoeren, welke als resultaat weer een binaire grootheid opleveren. Hieruit volgt, dat voor ons som en verschil van 2 binaire grootheden geen betekenis zullen hebben, tenzij in een formule als $\sum_{i=0}^{v-1} p_i^n f_i$, welke in het voorgaande werd afgeleid en waarbij uit de gedaante der p_i^n volgt, dat ten hoogste een der termen van nul verschillend kan zijn.

Ook aftrekken van de constante 1 is een uitzondering op deze regel. Producten hebben steeds betekenis, aangezien het product van 2 binaire grootheden steeds binair is. Aan quotiënten valt uiteraard geen betekenis toe te kennen, terwijl het verheffen van een binaire grootheid tot een macht welke voor iedere waarde combinatie der variabelen in de exponent een natuurlijk getal of nul is, natuurlijk toegelaten kan worden.

In dit laatste geval echter kunnen we dit natuurlijke getal steeds door 1 vervangen, aangezien k^n voor k binair en n natuurlijk getal steeds gelijk is aan k .

We zien dus dat onze pogingen er op gericht zullen moeten zijn de gegeven functies te schrijven in de vorm van producten van de standaardfuncties.

Beschouwen we nu het geval $n = 3$. We schrijven de gegeven functie $f(x,y,z)$ in de volgende vorm:

$$f(x,y,z) = (1-axy)(1-bxz)(1-cyz)(1-dx'y)(1-ex'z)(1-fy'z)(1-gxy') \times \\ \times (1-hxz')(1-iyz')(1-jx'y')(1-kx'z')(1-ly'z')(1-Axyz)(1-Bxyz') \times \\ \times (1-Cxy'z)(1-Dxy'z')(1-Ex'yz)(1-Fx'yz')(1-Gx'y'z)(1-Hx'y'z').$$

De gang van zaken is nu deze, dat we met behulp van het "lijstje" voor de functie f de in deze vergelijking voorkomende constanten bepalen.

Aangezien we voor $n = 3$ slechts 8 vergelijkingen vinden voor de constanten, gaat het er om een stel waarden voor $a \dots H$ te vinden, zodat aan alle vergelijkingen voldaan wordt. Hierbij zullen we om te beginnen al die coëfficiënten, welke onbepaald kunnen blijven $= 0$ stellen, aangezien de keuze 1 voor deze coëfficiënten uitsluitend tot gevolg zou hebben, dat nutteloos materiaal voor de verwezenlijking van de functie zou worden gebruikt.

Als verschillende keuzen van coëfficiënten combinaties aan de betrekkingen kunnen voldoen, zullen we die kiezen, bij welke het kleinste aantal variabelen optreedt, d.w.z. waarvoor

$$2(a+b+c+d+e+f+g+h+i+j+k+l) + 3(A+B+C+D+E+F+G+H)$$

minimaal is.

Deze vorm, waarin de functie daarna verschijnt, noemt Aiken de minimaal-vorm. Het kan natuurlijk voorkomen, dat er meer dan een minimaal-vorm mogelijk is.

Het plezierige van deze vorm is nu dat de "ontbinding" van de functie in T , P en C factoren nu onmiddellijk kan worden opgeschreven, immers een factor van de vorm $(1-xy)$ kan worden geschreven als

$P_2(x,y)$, terwijl een term $(1-xyz)$ kan worden vervangen door $T_1(\phi)$,
 $\phi = T_3(x',y',z')$.

Het onplezierige is echter, dat de aldus verkregen vorm meestal slechts dan aanleiding geeft tot een minimaal-schakeling als alle hoofdletterfactoren $= 0$ zijn. Is dit niet het geval, dan is het dikwijls mogelijk door verder aan de formules te scharrelen, een goedkoper resultaat te krijgen. Algemene regels voor deze verdergaande simplificaties zijn niet te geven en hoewel met de hier uitgestippelde methode, welke voor praktische doeleinden nog aanzienlijk vereenvoudigd kan worden, nuttige resultaten bereikt kunnen worden, ontberen we toch nog steeds de geruststellende zekerheid, dat we niet meer dure radiobuizen gebruikt hebben dan strikt noodzakelijk is.

Een ander, en nog klemmender bezwaar tegen de methode is, dat haar toepassing vrijwel geheel faalt wanneer men probeert haar toe te passen op meer gecompliceerde schakelementen.

Een element, dat in plaats van door de eenvoudige T, C of P functies bijvoorbeeld gekenmerkt zou worden door de functie:

$$f = x(yz + y'w)$$

dus

$$f = T_1(x') P_2(y,z') P_2(y',w')$$

zou in het hierboven geschetste schema nauwelijks passen, omdat het niet zo heel duidelijk is in welke vorm met onbepaalde coëfficiënten men de gegeven functie zou moeten schrijven om een ontbinding in deze vorm te krijgen.

Nog minder bruikbaar is de methode wanneer elementen met 2 uitgangen hun intrede zouden doen, hoewel dergelijke elementen zeer wel realiseerbaar en waarschijnlijk voor binaire schakeltechniek van eminent belang zouden zijn. Eén van de moeilijkheden is echter, zoals in het begin reeds werd opgemerkt, dat we momenteel niet in staat zijn welgefundeerde suggesties te doen omtrent de meest wenselijke vorm van de functie van een te construeren primair schakelement met bijv. 3 ingangen en 2 uitgangen, omdat de theorie ons op dit punt volkomen in de steek laat.

Ongeveer even onbevredigend is de situatie met betrekking tot die gevallen, waarin $m \neq 1$. Het beste wat men hier in het algemeen kan doen, is nagaan of de functies van elk der uitgangen afzonderlijk soms overeenkomstige factoren bevatten, waardoor een bepaalde hoeveelheid materiaal voor verschillende uitgangen dienst kan doen. Ook komt het nogal eens voor, dat de functies, welke de verschillende uitgangen afzonderlijk beschrijven, onderling een eenvoudig functioneel verband vertonen, waarvan men profijt kan trekken.

We zullen nu enkele eenvoudige circuits met $m \neq 1$ bespreken en wel in de eerste plaats het probleem der volledige of gedeeltelijke "decodatie" behandelen. Onder een decodator verstaan we daarbij een schakeling, waarvoor $n = p$ en $m = 2^p$ en welke zodanig is geconstrueerd, dat voor elk der 2^p mogelijke waardecombinaties der ingangen slechts 1 der uitgangen 1 is (voor verschillende ingangscombinaties verschillende uitgangen).

Vat men de p ingangen in de een of andere volgorde op als een binair geschreven natuurlijk getal, dan kan men dus bijvoorbeeld zeggen, dat als de ingang het getal k voorstelt, alleen de k^{de} uitgang = 1 is. De uitgangen hebben dus de functies:

$$\begin{aligned} f_0 &= x_1 x_2 \dots x_p \\ f_1 &= x_1' x_2 \dots x_p \\ f_2 &= x_1 x_2' \dots x_p \\ &\vdots \\ f_v &= x_1' x_2' \dots x_p' \quad v = 2^p - 1. \end{aligned}$$

Nemen we nu als voorbeeld bijv. $p = 4$, dan zien we dat

$$T_2[C_2(x,y), C_2(z,w)] = x'y'z'w'.$$

Een T_2 circuit kan dus een der gewenste uitgangen leveren mits de C_2 's gegeven zijn.

Dit betekent, dat we om alle uitgangen te vinden eerst alle C_2 circuits moeten maken. Dit zijn er in ons geval 8, nl.

$$\begin{array}{cccc} C_2(xy) & C_2(x'y) & C_2(xy') & C_2(x'y') \\ C_2(zw) & C_2(z'w) & C_2(zw') & C_2(z'w') \end{array}$$

De totale schakeling bevat dus 8 C_2 circuits + 16 T_2 's.

In het algemeen is voor de hoeveelheid materiaal, welke een schakeling gebruikt, de som der indices van de er in voorkomende T, C en P symbolen maatgevend: dit betekent technisch gezien nl. het aantal gebruikte roosters en is meestal equivalent aan het dubbele van het aantal gebruikte radiobuizen. In dit geval kunnen we dus de kosten van de schakeling stellen op $8 \times 2 + 16 \times 2 = 48$ totaal of 3 per uitgang.

De hier geschetste schakeling noemt men veelal een rechthoek wegens het uiterlijk dat hij aanneemt, als men hem in schemavorm op papier zet.

Het algemene voorschrift voor n ingangsvariabelen (n even verondersteld) luidt:

Verdeel de ingangsvariabelen in 2 gelijke groepen en decodeer deze volgens het geschetste systeem. Combineer vervolgens iedere uitgang van de ene groep met een van de andere groep met behulp van een T_2 circuit. Nadere beschouwing leert direct, dat de kosten per uitgang met toenemen de n naderen tot 2 ($n = 4 \rightarrow 3$

$$n = 5 \rightarrow 3$$

$$n = 6 \rightarrow 2.75$$

$$n = 7 \rightarrow 2.56$$

$$n = 8 \rightarrow 2.38).$$

Voor oneven n is de symmetrie niet meer geheel te handhaven en liggen de kosten, zoals uit het tabelletje ook blijkt, relatief iets ongunstiger.

Wanneer bepaalde uitgangen niet nodig zijn, leert inspectie van het schema direct, welke onderdelen kunnen worden weggelaten. De hier gemaakte kostenberekening kan nu dienen om althans voor iedere schake-

ling met $m \neq 1$ een bovengrens aan te geven voor de benodigde hoeveelheid onderdelen. Daartoe bedenken we, dat wanneer iedere uitgang afzonderlijk als functie van de n ingangsvariabelen in canonische vorm wordt neergeschreven, alle daarbij optredende termen door één rechthoek voor n variabelen kunnen worden geleverd. Daarna moet voor iedere uitgang nog gecombineerd worden, hetgeen met een C functie kan geschieden. Hierbij kunnen we dan nog bedenken, dat als het aantal termen voor een bepaalde uitgang $> v/2$ is, het aanbeveling verdient de "complementaire" termen te combineren met behulp van een T -functie. Zodoende vinden we dat de kosten voor een dergelijk circuit maximaal bedragen:

$$\sum_{j=1}^v M_j + g_n \text{ waarin:}$$

g_n = kosten van rechthoek voor n variabelen

$M_j = m_j$ of $v - m_j$ (welke de kleinste is)

m_j = aantal termen optredende in de canonische vorm der j^{de} uitgang.

In de praktijk zal het gewoonlijk niet verstandig zijn dit systeem te volgen, aangezien veelal door geschikte combinatie van onderdelen der voor de afzonderlijke uitgangen benodigde apparatuur een belangrijke kostenverlaging te bereiken zal zijn. Algemene regels hiervoor zijn echter helaas niet te geven en de vindingrijkheid en het opmerkingsvermogen van de constructeur blijven nog een (te) grote rol spelen.

CODERINGSPROBLEMEN

H.J.A. Duparc

Lezing in het colloquium Moderne Rekenmachines van het Mathematisch Centrum op 24 mei 1952 te Amsterdam.

§1. Inleiding

In het vervolg zullen wij elk der cijfers 0, 1, ..., 9 coderen door een vast aantal nullen en énen. Het is duidelijk dat dit vaste aantal n ten minste 4 moet zijn, want met 3 of minder cijfers 0 en 1 zijn ten hoogste 8 verschillende groeperingen te maken. Wij schrijven kortweg $x = (x_1, \dots, x_n)$. De getallen $x_1, \dots, x_n$ noemen wij de coördinaten van het getal x .

Neemt men $0 = (0, 0, 0, 0)$; $1 = (0, 0, 0, 1)$; $2 = (0, 0, 1, 0)$; $3 = (0, 0, 1, 1)$; ...; $9 = (1, 0, 0, 1)$, codeert men elk der getallen volgens de binaire schrijfwijze, dan heeft men voor $x = 0, 1, \dots, 9$

$$x = \sum_{n=0}^3 x_{n+1} 2^{3-n}.$$

De getallen $(x_1, \dots, x_n)$ zijn dan eenduidig door het getal x vastgelegd en omgekeerd. Wij zullen echter in het vervolg ook andere coderingen beschouwen dan degene, die corresponderen met de binaire schrijfwijze van het getal x .

Een getal in de decimale schrijfwijze coderen wij door elk zijner cijfers achtereenvolgens te coderen. Heeft zo'n getal m cijfers, dan bezit het derhalve mn coördinaten. Zelfs in het geval dat men voor n de minimale keuze $n = 4$ doet, heeft zo'n getal i.h.a. meer coördinaten, dan wanneer men het direct geheel in de duale schrijfwijze codeert.

Als twee natuurlijke getallen a en b gegeven zijn door hun coördinaten, is hun som ook bekend, zodat de coördinaten daarvan bepaald

zijn door die van de getallen a en b zelf. Hetzelfde geldt voor hun product. Beperken wij ons eerst tot het geval $0 \leq a \leq 9$, $0 \leq b \leq 9$ en onderstellen wij $a = (a_1, \dots, a_n)$; $b = (b_1, \dots, b_n)$. De som van a en b is dan gegeven door het aantal t zijner tientallen (dat 0 of 1 is) en het aantal s der eenheden. Elke coördinaat zowel van s als van t is dan een functie van de $2n$ getallen $a_1, \dots, a_n, b_1, \dots, b_n$. Wij schrijven

$$s_v = s_v(a_1, \dots, a_n, b_1, \dots, b_n);$$

$$t_v = t_v(a_1, \dots, a_n, b_1, \dots, b_n) \quad (v = 1, \dots, n),$$

ofwel kortweg

$$s = s(a, b); \quad t = t(a, b).$$

Zijn de functies s_v en t_v alle bekend, dan is hiermede ook door herhaaldelijk toepassen van deze formules de som van getallen van meer cijfers te vinden. Zo is b.v. voor de som efg van twee getallen ab en cd van twee cijfers (waarbij $a, \dots, g$ de cijfers der beschouwde getallen in de decimale schrijfwijze aangeven)

$$e = s(t(a, c), t(t(b, d), s(a, c))); \quad f = s(t(b, d), s(a, c)); \quad g = s(b, d).$$

Op geheel analoge wijze kunnen producten van getallen in decimale schrijfwijze gevonden worden zodra men behalve de opteloperaties ook kent de functies

$$p_v = p_v(a_1, \dots, b_n) \text{ en } q_v = q_v(a_1, \dots, b_n),$$

die ons van twee natuurlijke onder de 10 gelegen getallen a en b opleveren de codering der getallen p en q , die resp. aangeven het aantal eenheden en tientallen van het product ab .

Het is duidelijk dat de functies p_v , q_v , s_v en t_v afhangen van de wijze van codering der getallen $0, 1, \dots, 9$. Alvorens hierop nader in te gaan bewijzen wij eerst de volgende fundamentele stelling:

Iedere functie $f(x_1, \dots, x_N)$, die evenals elk der erin optredende variabelen $x_1, \dots, x_N$ slechts de waarden 0 en 1 aanneemt en die voor iedere mogelijke keuze der grootheden $x_1, \dots, x_N$ een voorgeschreven waarde aanneemt, is te schrijven in de gedaante

$$(1) \quad f(x_1, \dots, x_N) = \sum c_{v_1 v_2 \dots v_k} x_{v_1}^{a_1} x_{v_2}^{a_2} \dots x_{v_k}^{a_k};$$

in deze som loopt k van 0 tot N en bij zo'n k is $(v_1, \dots, v_k)$ een willekeurige combinatie van k der getallen $(1, \dots, N)$ en het geval $k = 0$ is bedoeld de ene term c op te leveren.

Het bewijs wordt gegeven door te laten zien dat de coëfficiënten c_v in het rechterlid van (1) zo gekozen kunnen worden dat voor iedere keuze van $(x_1, \dots, x_N)$ de functie $f(x_1, \dots, x_N)$ inderdaad de voorgeschreven waarde aanneemt. Het aantal onbekende coëfficiënten c_v is gelijk aan $\binom{N}{0} + \binom{N}{1} + \dots + \binom{N}{N} = 2^N$, terwijl het aantal opgelegde voorwaarden even groot is.

Substitutie van $(0, 0, \dots, 0)$ levert ons direct de waarde van c . Zij voorts voor $k = 0, 1, \dots, n$ reeds elke $c_{v_1 \dots v_k}$ gevonden. Elk der coëfficiënten $c_{v_1 \dots v_{n+1}}$ is nu te vinden door $x_{v_1} = \dots = x_{v_{n+1}} = 1$ te nemen en de overige variabelen x_v gelijk aan 0 te nemen. Men krijgt dan een vergelijking, waarin $c_{v_1 \dots v_{k+1}}$ optreedt, zodat uit deze betrekking de gewenste coëfficiënt onmiddellijk gevonden wordt. Hiermede is de fundamentele stelling bewezen.

Opmerkingen

In het geval dat slechts voor minder dan de 2^N mogelijke keuzen der grootheden $x_1, \dots, x_N$ de waarde der functie $f(x_1, \dots, x_N)$ voorgeschreven is, kan men voor de overige keuzen van $(x_1, \dots, x_N)$ de waarde der functie willekeurig voorschrijven, zodat dan tenminste één stel coëfficiënten c_v te vinden is waarvoor (1) geldt.

Ook in het geval dat elk der grootheden $x_1, \dots, x_N$ de p waarden $0, 1, \dots, p-1$ kan aannemen (waarbij p een willekeurig priemgetal is) en voor elk of voor een aantal van dergelijke keuzen van $(x_1, \dots, x_N)$ de functie $f(x_1, \dots, x_N)$ een voorgeschreven waarde aanneemt, die gelijk is aan 1 der getallen $0, 1, \dots, p-1$, bestaat er een analoge schrijfwijze

$$f(x_1, \dots, x_N) = \sum c_{a_1 a_2 \dots a_N} x_1^{a_1} x_2^{a_2} \dots x_N^{a_N},$$

waarin elk der grootheden $a_1, a_2, \dots, a_N$ gelijk is aan 0, 1, ... of $p-1$. De coëfficiënten c_a worden ook in dit geval bepaald door lineaire vergelijkingen, waarvan de coëfficiëntendeterminant gelijk blijkt te zijn aan

$((p-1)!!)^{Np^{N-1}}$; hierin is $(p-1)!! = 1!2! \dots (p-1)!$ en de betreffende coëfficiëntendeterminant is dus $\not\equiv 0 \pmod{p}$.

De fundamentele stelling voor functies is direct uit te breiden op stellen functies. Zij luidt dan: Ieder stel van m functies $f_1(x_1, \dots, x_N), \dots, f_m(x_1, \dots, x_N)$, die evenals de grootheden $x_1, \dots, x_N$ slechts de waarden 0 en 1 aannemen en voor iedere of sommige van alle mogelijke keuzen der grootheden $(x_1, \dots, x_N)$ voorgeschreven waarden (0 of 1) aannemen, is van de gedaante

$$f_\mu(x_1, \dots, x_N) = \sum c_{\mu v_1 \dots v_k} x_1^{v_1} \dots x_k^{v_k} \quad (\mu = 1, \dots, m),$$

waarbij het somteken weer een analoge betekenis heeft als in (1).

In het geval dat $m = N$ is, leveren de functies ons uit een stel van N grootheden $(x_1, \dots, x_N)$ een nieuw stel van N grootheden $(y_1, \dots, y_N)$. Wij schrijven dit in operatorvorm $y = Fx$.

Op de bovenbeschouwde functies p_v, q_v, s_v en t_v is het zojuist gevondene van toepassing. Men heeft daarbij $N = 2n$; $m = n$.

Wij beschouwen thans de functies p_v en s_v eens nader. Op grond van het gevondene is elke functie p_v bepaald door de keuze der codering van elk der cijfers 0, 1, ..., 9. Dat is in het bijzonder het geval met de functies $s_v(a_1, \dots, a_n, b_1, \dots, b_n)$ of kortweg $S(a, b)$ in het geval $b = 1$ is. Deze formules geven ons de codering van $a+1$ uit die van $a \pmod{10}$. Geven wij ze aan met S_1 , dan hebben wij dus $a + 1 = S_1 a = S(a, 1)$; $a + b = S_1^b a = S(a, b)$, waarvoor wij ook wel zullen schrijven $S_b a$. Wij merken op dat de operatoren S_b een groep vormen. Men heeft nl. $S_b S_c = S_d$, waarbij $d \equiv b + c \pmod{10}$ en $0 \leq d \leq 9$. De groep dezer ope-

ratoren is isomorph met de additieve groep der restklassen $K_0, \dots, K_9$ mod 10. Evenzo heeft men voor de verzameling der operatoren P_b , waarbij onder $c = P_b a$ wordt verstaan $c_v = p_v(a_1, \dots, a_n, b_1, \dots, b_n)$ voor $v = 1, \dots, n$, dat $P_u P_v = P_w$ dan en slechts dan als $K_u K_v = K_w$. De verzameling der operatoren P_b vormt dus evenmin als die der restklassen K_b een multiplicatieve groep.

§2. Keuze der codering

Door de codering der getallen 0, 1, ..., 9 te geven zijn de operatoren S_b en P_b zoals wij al opmerkten alle bepaald. Dit is ook het geval als men geeft de codering van een der getallen 0, 1, ..., 9 en ook de operator S_1 (of S_3, S_7 of S_9), waarbij echter de operator S_1 de orde 10 (en geen lagere orde) moet bezitten, d.w.z. $S_1^{10} = S_0$ en $S_1^g \neq S_0$ voor $g = 1, \dots, 9$.

Wij merken nog op dat de orde der operatoren S_3, S_7 en S_9 ook 10 is, dat die van elk der operatoren S_2, S_4, S_6 en S_8 gelijk is aan 5 en de orde van S_5 gelijk is aan 2.

De operatoren P_3, P_7 en P_9 bezitten de orde 4, terwijl bij de operatoren P_0, P_2, P_4, P_5, P_6 en P_8 feitelijk van geen orde sprake is.

In de praktijk is het nuttig een codering zo te kiezen dat de operatoren S_b en P_b alle zo eenvoudig mogelijk worden. Een bijzonder eenvoudige keuze van een operator S_b zou die zijn waarbij deze slechts de coördinaten permuteert. Allereerst houdt dat in dat bij elk der cijfers 0, 1, ..., 9 een gelijk aantal van de coördinaten gelijk is aan 0. Zij dit aantal h , dan moet dus $\binom{n}{h} \geq 10$ zijn, dus $n \geq 5$. Wij kunnen echter meer zeggen, omdat die operator S_1 de orde 10 hebben moet. Is S_1 slechts een permutatie der n coördinaten, dan bezit S_1 een orde die gelijk is aan het K.G.V. der getallen $n_1, n_2, \dots, n_k$, waarbij $n_1 + n_2 + \dots + n_k = n$.

Bijgevolg is pas bij $n = 7$ een opteloperator mogelijk, die de coördinaten der gecodeerde getallen permuteert. Wij zoeken thans echter naar weliswaar minder eenvoudige operatoren, waarbij evenwel het aantal der vereiste coördinaten kleiner kan zijn.

Nog op betrekkelijk eenvoudige wijze te realiseren zijn de groot-heden $x_1 x_2$ en $(x_1' x_2')'$. Hierin wordt onder x_1' verstaan $1 - x_1$. Deze zijn door dioden te verwezenlijken terwijl de grootheden $x_1' x_2' \dots x_k'$, $(x_1' x_2' \dots x_k')'$ en $(x_1 x_2)'$ een ingewikkelder apparatuur ¹⁾ vereisen (resp. triode, cathode-volger en pentode), waarbij k willekeurig is.

Wij hebben dus thans na te gaan of er optel- en vermenigvuldigopera-toren zijn te vinden in de gevallen dat men uitsluitend dioden toelaat of dat men ook nog de ingewikkelder apparatuur nodig heeft. Bij $n \geq 7$ be-hoeft, zoals wij zagen voor additie van 1 zelfs geen diode gebruikt te worden. Bij $n = 7$ kan men de algemene additieoperatoren inderdaad door dioden verwezenlijken. Neemt men nl.

$$\begin{aligned} 0 &= (0, 0, 0, 0, 0, 1, 0, 1) \\ 1 &= (1, 0, 0, 0, 0, 0, 1, 0) \\ 2 &= (0, 1, 0, 0, 0, 0, 0, 1) \\ 3 &= (0, 0, 1, 0, 0, 0, 1, 0) \\ 4 &= (0, 0, 0, 1, 0, 0, 0, 1) \\ 5 &= (0, 0, 0, 0, 1, 1, 0, 0) \\ 6 &= (1, 0, 0, 0, 0, 0, 0, 1) \\ 7 &= (0, 1, 0, 0, 0, 0, 1, 0) \\ 8 &= (0, 0, 1, 0, 0, 0, 0, 1) \\ 9 &= (0, 0, 0, 1, 0, 1, 0, 0) \end{aligned}$$

dan is S_1 de operator

$$\begin{aligned} f_1(x) &= x_5, \quad f_2(x) = x_1, \quad f_3(x) = x_2, \quad f_4(x) = x_3, \quad f_5(x) = x_4, \\ f_6(x) &= x_7, \quad f_7(x) = x_6. \end{aligned}$$

Men vindt dan voor $s = x + y$ de algemene optelformules

¹⁾ Verg. de vorige voordracht in deze serie: Logische synthese van reken-circuits, door B.J. Loopstra, Rapport ZW 1952-010.

$$s_1 = x_1 y_5 + x_2 y_4 + x_3 y_3 + x_4 y_2 + x_5 y_1$$

$$s_2 = x_1 y_1 + x_2 y_5 + x_3 y_4 + x_4 y_3 + x_5 y_2$$

$$s_3 = x_1 y_2 + x_2 y_1 + x_3 y_5 + x_4 y_4 + x_5 y_3$$

$$s_4 = x_1 y_3 + x_2 y_2 + x_3 y_1 + x_4 y_5 + x_5 y_4$$

$$s_5 = x_1 y_4 + x_2 y_3 + x_3 y_2 + x_4 y_1 + x_5 y_5$$

$$s_6 = x_6 y_7 + x_7 y_6$$

$$s_7 = x_6 y_6 + x_7 y_7$$

en op grond van de grondstelling voor operatoren zijn ook de formules voor $t(x,y)$, $p(x,y)$ en $q(x,y)$ gemakkelijk te vinden.

De hier optredende formules (en ook die voor de 3 genoemde overige operatoren) hebben de eigenschap, dat er slechts één term in optreedt, die van nul verschilt. Terwijl in het algemeen voor het vinden van de som $x_i + x_j \pmod{2}$ uit x_i en x_j trioden vereist zijn, is dit bij een som met deze bijzondere eigenschap niet zo en kan men daarbij met dioden toe.

Bij onze codering waarbij weliswaar de operaties eenvoudig verlopen, maar het aantal variabelen vrij groot is, bestaat tussen de coördinaten der getallen de relatie $\sum_{i=1}^5 x_i = x_6 + x_7 = 0$, zodat x_5 en x_7 overtallige coördinaten zijn. Wij kunnen dus ook werken met een codering waarbij $n = 5$ is en waarin men heeft

$$0 = (0, 0, 0, 0, 0)$$

$$1 = (1, 0, 0, 0, 1)$$

$$2 = (0, 1, 0, 0, 0)$$

$$3 = (0, 0, 1, 0, 1)$$

$$4 = (0, 0, 0, 1, 0)$$

$$5 = (0, 0, 0, 0, 1)$$

$$6 = (1, 0, 0, 0, 0)$$

$$7 = (0, 1, 0, 0, 1)$$

$$8 = (0, 0, 1, 0, 1)$$

$$9 = (0, 0, 0, 1, 1).$$

Het nadeel hierbij is dat men bij de somformules hierboven ook reeds b.v. bij s_1 de beschikking moest hebben over de grootheden x_5 en y_5 , maar deze moeten nu gevonden worden uit $1 - x_1 - x_2 - x_3 - x_4$ resp. $1 - y_1 - y_2 - y_3 - y_4$, hetgeen meer dan alleen dioden vereist.

Inderdaad vindt men in het nieuwe systeem waarvan wij de coördinaten van het getal x in verband met het vorige aangeven met $(x_1, x_2, x_3, x_4, x_6)$ voor de som $s = x + y$ dat

$$s_6 = x_6(1 - y_6) + (1 - x_6)y_6 \equiv x_6 + y_6$$

zodat trioden enz. onontbeerlijk zijn.

Tenslotte onderzoeken wij het geval $n = 4$ nader en laten zien dat het gebruik van uitsluitend dioden bij de optelling bepaalde coderingen uitsluit. Beschouw de operator S_1 , die aan ieder cijfer zijn opvolger toevoegt, d.w.z. onderzoek de formules

$$y_h = s_h(x_1, x_2, x_3, x_4) \quad (h = 1, 2, 3, 4),$$

waarbij $y = x + 1$ is. Elke formule $s_h(x_1, x_2, x_3, x_4)$ is volgens onze grondstelling een polynoom, dat opgebouwd is uit één of meer der termen $x_i, x_i x_j, x_i x_j x_k, x_1 x_2 x_3 x_4$. Laten wij slechts de productoperatoren $x_i x_j$ toe, dan mag zo'n som slechts dan uit meer termen bestaan als die niet tegelijkertijd gelijk zijn aan 1, d.w.z. als hun product nul is. Daar met dioden ook nog de operator $x_i \cdot x_j = (x_i' x_j')$ te verwezenlijken is, mag onze som ook hieruit op te bouwen termen bevatten mits alweer niet tegelijkertijd twee der termen gelijk zijn aan 1. Zoekt men alle producten van 4 coördinaten a, b, c en d uit, die te verwezenlijken zijn met dioden, dan vindt men onderstaande mogelijkheden of degene, die er uit door permutatie der grootheden x_1, x_2, x_3 en x_4 te vinden zijn. Men lette er hierbij op dat voor gedurige producten waarin zowel de elementaire als de met een nulletje aangegeven vermenigvuldiging de associatieve wet niet geldt. In de producten is de volgorde der bewerkingen de natuurlijke aangegeven door de volgorde der factoren. Men berekene eerst het product

der eerste twee factoren, vermenigvuldigde dit met de derde enz., terwijl slechts haakjes hierin wijziging brengen.

$$\begin{aligned}
 & x_1 \\
 & x_1 x_2 \\
 & x_1 \cdot x_2 = x_1 x_2 + x_1 + x_2 \\
 & x_1 x_2 x_3 \\
 & x_1 x_2 \cdot x_3 = x_1 x_2 x_3 + x_1 x_2 + x_3 \\
 & x_1 \cdot x_2 x_3 = x_1 x_2 x_3 + x_1 x_3 + x_2 x_3 \\
 & x_1 \cdot x_2 \cdot x_3 = x_1 x_2 x_3 + x_1 x_2 + x_1 x_3 + x_1 + x_2 + x_3 \\
 & x_1 x_2 x_3 x_4 \\
 & x_1 x_2 x_3 \cdot x_4 = x_1 x_2 x_3 x_4 + x_1 x_2 x_3 + x_4 \\
 & x_1 x_2 \cdot x_3 x_4 = x_1 x_2 x_3 x_4 + x_1 x_2 x_4 + x_3 x_4 \\
 & x_1 x_2 \cdot x_3 \cdot x_4 = x_1 x_2 x_3 x_4 + x_1 x_2 x_3 + x_1 x_2 x_4 + x_1 x_2 + x_3 x_4 + x_3 + x_4 \\
 & x_1 \cdot x_2 x_3 x_4 = x_1 x_2 x_3 x_4 + x_1 x_3 x_4 + x_2 x_3 x_4 \\
 & x_1 \cdot x_2 x_3 \cdot x_4 = x_1 x_2 x_3 x_4 + x_1 x_2 x_4 + x_1 x_3 x_4 + x_2 x_3 x_4 + x_1 x_4 + x_2 x_4 + x_3 x_4 \\
 & x_1 \cdot x_2 \cdot x_3 x_4 = x_1 x_2 x_3 x_4 + x_1 x_2 x_4 + x_1 x_3 x_4 + x_2 x_3 x_4 + x_1 x_4 + x_2 x_4 + x_3 x_4 \\
 & x_1 \cdot x_2 \cdot x_3 \cdot x_4 = x_1 x_2 x_3 x_4 + \sum x_1 x_2 x_3 + \sum x_1 x_2 + \sum x_1 \\
 & x_1 x_2 \cdot (x_3 x_4) = x_1 x_2 x_3 x_4 + x_1 x_2 + x_3 x_4 \\
 & x_1 x_2 \cdot (x_3 \cdot x_4) = x_1 x_2 x_3 x_4 + x_1 x_2 x_3 + x_1 x_2 x_4 + x_1 x_2 + x_3 x_4 + x_3 + x_4 \\
 & x_1 \cdot x_2 (x_3 \cdot x_4) = x_1 x_2 x_3 x_4 + \sum x_1 x_2 x_3 + x_1 x_3 + x_1 x_4 + x_2 x_3 + x_2 x_4.
 \end{aligned}$$

In geval bij deze producten twee of meer der factoren samenvallen, leveren zij geen uitdrukking op van een andere structuur dan de reeds opgesomde.

Elke formule $s_h(x_1, x_2, x_3, x_4)$ ($h = 1, 2, 3, 4$) deel uitmakende van de operator S_1 bestaat dan uit de som van een of meer van de opgesomde termen of van de daaruit door permutatie der grootheden x_1, x_2, x_3, x_4 verkregen termen.

Het is duidelijk dat geen der getallen 0, 1, ..., 9 een codering (0, 0, 0, 0) kan bezitten want wegens $00 = 0$ en $0 \cdot 0 = 0$ zou dan elk product ook 0 zijn en de operator S_1 , die uit deze producten is opgebouwd, zou aan het getal (0, 0, 0, 0) weer dit getal toevoegen in strijd met de betekenis van S_1 . Omdat $1 \cdot 1 = 1$ en $1 \cdot 1 = 1$ is zou evenzo de operator S_1

aan het getal (1, 1, 1, 1) weer het getal (1, 1, 1, 1) toevoegen, wat evenmin mogelijk is, zodat ook geen der getallen de codering (1, 1, 1, 1) mag bezitten. In de codering van elk getal treden dus zowel cijfers 0 als cijfers 1 op.

Welke mogelijkheden nu open blijven dient nader te worden onderzocht. Om tenslotte een denkbeeld te geven hoe men dit onderzoek kan doen verlopen, beschouwen wij de codering, waarbij 4 getallen gecodeerd worden met 3 coördinaten 0 en 1 coördinaat 1, terwijl de overige 6 getallen alle met 2 cijfers 0 en 2 cijfers 1 worden gecodeerd. Wij leggen daarbij aan de structuur der formules S_1 een verder gaande beperking op door te eisen dat alle erin optredende producten slechts uit twee factoren bestaan. Deze beperking komt voort uit praktische overwegingen. Bij producten van meer dan 2 factoren is nl. een extra apparatuur nodig om de stromen waarvoor de coördinaten worden gerepresenteerd te versterken. Bij een product van twee factoren is dit nog niet nodig.

In dit speciale geval treden slechts als bouwstenen der formules S_1 op de 3 producten x_i , $x_i x_j$ en $x_i \cdot x_j$. Letten wij nog op de eis dat geen $s_h(x_1, x_2, x_3, x_4)$ mag bestaan uit twee termen, die voor een optredende keuze van (x_1, x_2, x_3, x_4) beide gelijk zijn aan 1, dan is voor elke s_h slechts het volgende zevental formules mogelijk

$$x_i; x_i + x_j x_k; x_i + x_j x_k + x_l x_1; x_i + x_j x_k + x_l x_1 + x_l x_j; \sum_n x_i x_j; x_i \cdot x_j; x_i \cdot x_j + x_l x_1.$$

Hierin zijn de getallen i, j, k en l twee aan twee verschillend en $\sum_n$ wordt uitgestrekt over n verschillende termen van de gedaante $x_i x_j$; uiteraard is daarbij $1 \leq n \leq 6$. Men kan nu aantonen dat de bovengenoemde codering dan onmogelijk is. Immers bij 4 der 10 getallen is $x_1 = 1$. Wij gaan dus na of inderdaad ook bij b.v. de formule $s_1(x_1, x_2, x_3, x_4) = x_i + x_j x_k$ voor precies 4 der getallen (x_1, x_2, x_3, x_4) de waarde van s_1 gelijk is aan 1. Nu is $s_1 = 1$ als $x_i = 1$ en $x_j x_k = 0$. Uit $x_i = 1$ volgt automatisch $x_j x_k = 0$, dus reeds in de vier gevallen dat $x_i = 1$ is, is $s_1 = 1$. Als echter $x_i = 0$ is, is $s_1 = 1$ in het enige geval dat $x_j = x_k = 1$, zodat nu 5 getallen x de eigenschap zouden hebben dat hun opvolger een 1^e coördi-

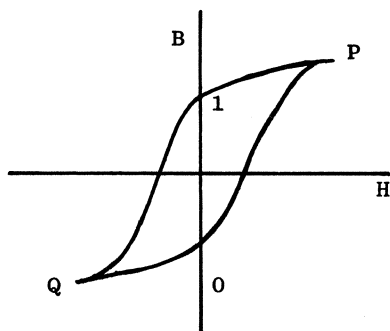
naat 1 bezit, wat echter niet het geval is. Op analoge wijze sluit men het optreden van de andere mogelijkheden voor s_1 uit, afgezien van $s_1 = x_i$, maar dit leidt tot het reeds eerder besproken geval van permutatie, dat eveneens niet kan optreden.

HET GEBRUIK VAN FERROMAGNETISCHE MATERIALEN IN REKENCIRCUITS

B.J. Loopstra

Lezing in het colloquium Moderne Rekenmachines van het Mathematisch Centrum op 24 mei 1952 te Amsterdam.

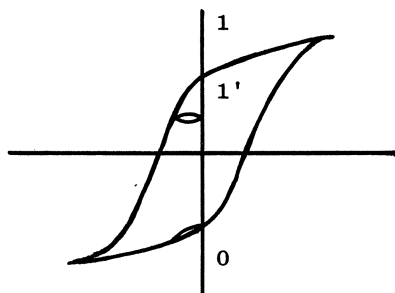
1. De meest voor de hand liggende eigenschap van permanent ferromagnetisch materiaal welke van nut kan zijn voor logische schakelementen, is die der magnetische verzadiging: elke magnetische kern kan zich in 2 verzadigingstoestanden bevinden, welke we in de B-H-karakteristiek zullen aangeven als 0 en 1. Een dergelijke kern kan dus wor-



den gebruikt als onthoud-element voor een enkele binaire digit. Onder de verzadigingstoestand verstaan we hierbij dus die punten op de hysteresekromme, waarvoor $H = 0$, welke punten bij de meeste materialen gekenmerkt worden door een vrij wat lagere $|B|$ dan de punten P en Q, waar $1-dB/dH$ verwaarloosbaar begint te worden. Bij enkele speciale alliages echter, waarbij de hystereselus bijna rechthoekig verloopt, verschilt B_1 slechts ca. 3% van B_P . De eerste toepassing van kernen met dit materiaal is geweest de constructie van statische magnetische vertragslijnen.

Bij het gebruik van een magnetische kern als onthoud-element doen zich 2 problemen voor: het opnemen van de gewenste informatie (schrijven) en het afgeven ervan (lezen). In het algemeen is het tweede probleem het lastigste. Het eerste is nl. eenvoudig te realiseren door het aanbrengen van een magnetiserende kracht $\pm H$ van vol-

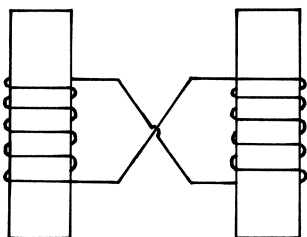
doende amplitude. Bij het tweede probleem wordt echter geëist dat de leesoperatie de in de kern vertegenwoordigde informatie intact zal laten en diensgevolge is het een vrij hachelijke onderneming de staat van de kern af te lezen door bijv. een kleine magnetiserende kracht $-\Delta H$ aan te brengen en te zien hoe de kern hierop reageert. Immers, bevatte de kern oorspronkelijk een 0, d.w.z. bevond hij zich



in punt 0 van de hystereselus, dan zal hij na afloop van het toepassen van een magnetiserende kracht $-\Delta H$ naar het punt 0 terugkeren. Bevatte de kern echter een 1, dan gaat het materiaal naar punt 1'. Hoewel dus de eerste keer, zoals uit het figuurtje

blijkt, de ΔB in de 1-stand veel groter is dan in de 0-stand, zal dit de 2e en volgende keren niet meer het geval zijn, zodat we de stand van het element slechts 1 keer zouden kunnen lezen. Hoewel men kan zeggen dat de informatie, welke de kern bevat, behouden blijft, komt deze na een maal lezen in een zodanige vorm, dat het aflezen ervan onmogelijk wordt.

Om aan deze moeilijkheid te ontkomen, kan men het lezen als een quasi-schrijven uitvoeren. Daartoe past men een grote negatieve ΔH toe, welke de kern beslist in stand 0 brengt. De flux variatie $1 \rightarrow 0$ is veel groter dan $0 \rightarrow 0$ en aflezen is eenvoudig. Nu moet men echter in het geval, dat een 1 uitgelezen wordt, de oorspronkelijke toestand weer herstellen. Dit kan in principe op verschillende manieren gebeuren. Van belang hierbij is dat de fluxvariatie $1 \rightarrow 0$ voldoende is om met behulp van een hulpwikkeling een andere kern in verzadiging te



drijven, zodat men twee kernen, welke aanvankelijk resp. in stand 1 en 0 waren, na het aflezen in stand 0 resp. 1 aantreft. Men kan dus de oorspronkelijke toestand herstellen door nu nog eens de

tweede kern "af te lezen" zonder dat men van het uitgelezen resultaat gebruik maakt. Hetzelfde resultaat zou te bereiken zijn door slechts een kern te gebruiken en na afloop van de leesoperatie direct de gelezen informatie terug te schrijven, een systeem dat echter voor de toepassing in vertragslijnen minder aanbevelenswaardig lijkt.

Voor de technische uitvoering der vertragslijnen zie verder (1).

De laatste tijd is door Forrester (2) en Papian (3) gewezen op de mogelijkheid dit type kernen te gebruiken voor het snelle geheugen van elektronische rekenmachines. Dergelijke geheugens bieden in principe twee problemen:

- 1e. het onthouden van een grote hoeveelheid digits
- 2e. het snelle bereikbaar zijn ervan.

Het 2e probleem, het selectieprobleem, kan in principe op twee manieren worden aangepakt. Men kan, en dat zou verreweg de prettigste methode zijn, het zo inrichten dat iedere digit, bijv. door het instellen van een bepaalde selectiecode, vrijwel instantaan gelezen of geschreven kan worden. Ook kan men de informatie op de een of andere manier continu langs een of meer aftast-punten laten lopen, zodat men wachten moet tot de gewenste digits beschikbaar zijn (vertragslijnen, magnetische trommels). Het eerstgenoemde codesysteem zou te verwezen-

	A	B	C
a	1	2	3
b	4	5	6
c	7	8	9

lijken zijn, indien men een element kon vinden, dat slechts bij "bekrachtiging" van bv. 2 ingangslijnen een uitgangsvariatie zou opleveren. Immers dan kan men met behulp van twee n-punts codes n^2 elementen selecteren. Men kan het zo beschouwen, dat de elementen zich bevinden op de rooster-

punten van een rechthoekig coördinatensysteem, zodat door keuze van een x en een y coördinaat juist een roosterpunt kan worden uitgekozen. In de figuur komt code B_c met element 8 overeen.

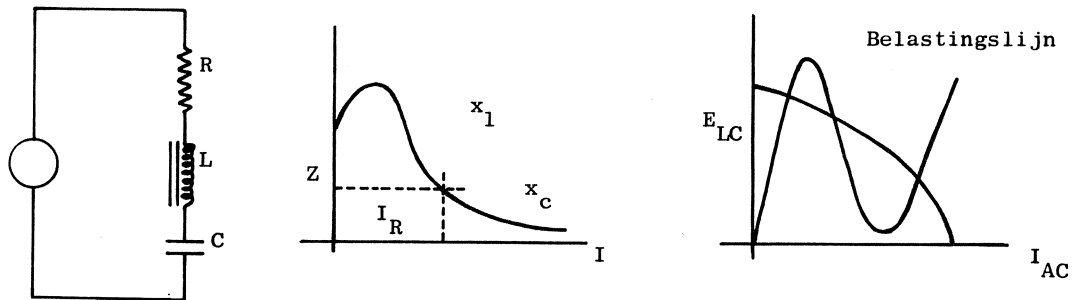
Wanneer nu als element een magnetische kern wordt gekozen met 2 selectiewikkelingen, dan is het mogelijk de afleessignalen zo te kiezen, dat bekrachtiging van een wikkeling slechts een kleine fluxvariatie doet ontstaan, terwijl bekrachtiging van beide wikkelingen gelijktijdig in het geval van een 0 een grote en in het geval van een 1 een kleine variatie doet ontstaan.

Wat betreft de moeilijkheden verbonden aan het gebruik van selectiesignalen van geringe amplitude, waarop in het voorafgaande reeds werd gewezen, zie men verder de genoemde literatuur.

We kunnen nog opmerken, dat het ook denkbaar zou zijn slechts een selectiewikkeling te gebruiken en daarvan beide "einden" te selecteren, m.a.w. door selectie een gesloten circuit, dat de selectiespoel bevat, te laten ontstaan en vervolgens met een grote ΔH af te lezen. Dat een dergelijk systeem met relais uitvoerbaar is, is duidelijk. Ook met elektronische schakelapparatuur zou het wellicht te verwezenlijken zijn.

Tenslotte willen we nog even nagaan in hoeverre dergelijke snelle geheugens economisch verantwoord geacht kunnen worden. Een magnetische kern van speciaal materiaal, met verscheidene wikkelingen, adaequaat gemonteerd, zal waarschijnlijk op een prijs van ca. f 10,- per stuk komen. Een geheugen van 1024 woorden van bv. 30 digits dus op een totale prijs van ruim 3 ton! Vergelijking met bijvoorbeeld een geheugen met kwikvertragslijnen levert dus ongeveer een factor 6 hogere prijs. Hiertegenover staat dat men een veel kleinere wachttijd kan bereiken en dat elk magnetisch geheugen permanent is, dus tot op zekere hoogte immuun tegen het uitvallen van voedingsspanningen e.d.

2. Behalve permanent ferromagnetisch materiaal, is ook niet-permanent materiaal wellicht bruikbaar voor verscheidene toepassingen. Beschouwen we in de eerste plaats weer het probleem van het onthoud-element, dan valt de aandacht op de in (4) beschreven ferro-resonante flip-flop.



Indien L een zelfinductie met ijzerkern is, kan de amplitude van de aangelegde wisselspanning zo worden gekozen, dat althans gedurende een gedeelte van de cyclus verzadiging in het ijzer optreedt. Dit betekent, dat de effectieve waarde van L verkleind wordt en men krijgt een grafiek van $L - I$ zoals aangegeven in de figuur. Hieruit volgt een $E_{LC} - I_{AC}$ karakteristiek uit de 2e figuur. Het blijkt dat een stuk met negatieve weerstand optreedt, zodat een belastingslijn zo kan worden gekozen, dat het circuit twee stabiele posities heeft.

Het blijkt mogelijk de frequentie van de wisselspanning op te voeren tot enkele Megacycles en op deze manier omslaan van het element van de ene stand naar de andere mogelijk te maken in ca. $10 \mu\text{sec}$. Een flip-flop van dit type gebruikt in een technisch bruikbare uitvoering twee kernen en heeft, door het reactieve karakter van de elementen, een zeer laag energieverbruik. Verder heeft hij met elektronische flip-flops, in tegenstelling tot permanent-magnetische, gemeen, dat voortdurend energie afgegeven kan worden. Hoewel we hier dus te maken hebben met een element, dat essentieel op wisselspanning gebaseerd is, kan het door gelijkrichting van de uitgang ook in circuits, welke op gelijkspanning gebaseerd zijn, dienst doen en daarbij, door de hoge frequentie, welke de wisselspanning heeft, nog betrekkelijk hoge snelheden bereiken.

Met de hier geschetste mogelijkheden in gedachte, kan men zich afvragen in hoeverre bijv. ook de langzamerhand vrij ver gevorderde techniek der magnetische versterkers mogelijkheden biedt voor de constructie van schakelementen met operatiesnelheden, welke weliswaar

niet kunnen concurreren met die van electronische apparatuur, maar die wellicht door betrouwbaarheid van werking en gering energieverbruik toch nog wijde toepassingsmogelijkheden zouden bieden.

Het feit, dat men bijv. met behulp van een DC signaal door een wikkeling op een kern de zelfinductie hiervan kan wijzigen, waardoor men hetzij de amplitude van een AC signaal door een tweede wikkeling kan beïnvloeden, hetzij het harmonische gehalte ervan kan wijzigen en amplitude discriminatie kan verkrijgen door het gebruikmaken van geschikte filters, biedt zeker mogelijkheden tot de constructie van "poorten" voor wisselspanningssignalen. Als bij deze elementen de basisfrequentie voldoende kan worden opgevoerd, kan de snelheid een redelijke waarde bereiken. Verder onderzoek in deze richting lijkt in elk geval zeer gewenst.

Literatuur

(1) Static Magnetic Storage and Delay Line.

An Wang and Way Dong Woo

Journal of Applied Physics, vol. 21, no. 1,
pp. 49-54, January 1950.

(2) J.W. Forrester, Digital information storage in three dimensions
using magnetic cores.

Journal of Applied Physics, vol. 22, pp. 44-48,
January 1951.

(3) William N. Papian, A coincidence-current magnetic memory cell
for the storage of digital information.

Proceeding of the I.R.E., vol. 40, no. 4,
pp. 475-478, April 1952.

(4) Carl Isborn, Ferroresonant flip-flops.

Electronics, April 1952, pp. 121-123.

WELKE OPDRACHTEN ZIJN ESSENTIEEL IN EEN REKENMACHINE?

W.L. van der Poel

Lezing in het colloquium Moderne Rekenmachines van het Mathematisch Centrum op 27 juni 1952 te Amsterdam.

Daar velen van U reeds in de gelegenheid zijn geweest om nader kennis te maken met moderne automatische rekenmachines, zal ik niet spreken over deze machine zelf, doch uitsluitend over de opdrachten die zo'n machine moet kunnen uitvoeren. Een veel verbreide opvatting is o.a. dat een rekenmachine minstens een soort geconditioneerde opdracht moet hebben om de loop van zijn berekening zelf te kunnen kiezen. In deze lezing hoop ik aan te tonen, dat dit niet het geval is.

Om te beginnen wil ik enkele veel gebruikte termen definiëren en moet ik om de gedachten te bepalen iets over de machine vertellen. We nemen aan te doen te hebben met een machine met een één-adrescode. Elke instructie uit het program bestaat dan uit een adres - het numerieke deel van de instructie, dat zegt waar het getal waar de operatie op verricht moet worden te vinden is in het geheugen - en een opdracht, die zegt wat er met dat getal gedaan moet worden. De machine bestaat uit een accumulator A, een geheugen G, dat een groot aantal registers bevat en een besturing. Verder nemen wij aan dat de machine inwendig in het tweetallig stelsel werkt. Voor de representatie van negatieve getallen zullen complementen worden gebruikt. De machine zal de volgende opdrachten verstaan:

Optellen zonder schoonmaken van de accumulator O(pt)
 Aftrekken zonder schoonmaken van de accumulator A(ftr)
 Opbergen met schoonmaken van de accumulator S(ubst)
 Opbergen zonder schoonmaken van de accumulator R
 Een ongeconditioneerde sprongopdracht X
 Een geconditioneerde opdracht C(ond).

Deze geconditioneerde opdracht werkt zo, dat als de inhoud van de accumulator negatief is, normaal wordt doorgegaan, maar als de inhoud van de accumulator 0 is, dat dan de eerstvolgende instructie overgeslagen wordt. Dan zal de machine nog opdrachten hebben voor vermenigvuldigen, delen, schuiven naar links en rechts, een invoeropdracht, een uitvoeropdracht, een stopopdracht en nog eventueel een opdracht voor afronden, logische vermenigvuldiging, enz.

We zullen beginnen aan te tonen, dat de opdracht schuiven naar links misbaar is. Schuiven naar links is in het tweetallig stelsel verdubbelen, en verdubbelen is optellen van een getal bij zichzelf. Hiermee is deze opdracht tot optellen teruggevoerd.

Vervolgens is de stopopdracht onnodig. Om aan het einde van een program geen essentiële handelingen meer te laten verrichten, kan men de machine laten "cirkelen", d.w.z. met een instructie X n in register n kunnen we zorgen, dat wanneer het program eenmaal op dit register aankomt, steeds weer de inhoud van dit register uitgevoerd wordt.

De invoer- en uitvoeropdracht kunnen we kwijtraken door een tweetal speciale registers te installeren. Het ene register is zo ingericht, dat met het aanhalen van een getal uit dit register een nieuw teken uit de buitenwereld wordt opgenomen, terwijl het andere register bij het opbergen van een getal in dit register, dit getal uitvoert. Daarmee zijn dan in- en uitvoer teruggebracht tot A en S opdrachten.

De rekenkundige elementen waaruit vermenigvuldigen bestaat zijn: optellen en schuiven naar links. We kunnen voor een vermenigvuldiging dus een program maken waarin uitsluitend optellingen en geconditioneerde opdrachten e.d. voorkomen. Voor het geval dat het product van twee getallen de capaciteit van een register niet overschrijdt, kunnen wij bijvoorbeeld het volgende program gebruiken:

0	0	Benodigde constanten
1	1	
2	a	Vermenigvuldiger wordt hier geplaatst voor de aanvang

	3	b		Vermenigvuldigtal wordt hier geplaatst voor de aanvang
	(4	prod)		Register gereserveerd voor product
	(5	telling)		Register gereserveerd voor de telling waardoor het eind van de bewerking geïndiceerd wordt
→	6	A 1		Startpunt. Neem als telling -1 = 11111...11 op
	7	S 5		Plaats telling in register 5
	8	S 4		} Maak register voor product schoon
21 →	9	O 2		
	10	O 2		
	11	R 2		
	12	C		Onderzoek het meest linkse cijfer van de vermenigvuldiger
	13	O 3		} Indien 1 dan vat C het getal negatief op en wordt het vermenigvuldigtal opgeteld
	14	O 4		
	15	O 4		
	16	S 4		en plaats dit gedeeltelijk product weer in 4
	17	O 5		} Verdubbel daarna de telling welke dus rechts een nul meer krijgt en links een een minder
	18	O 5		
	19	R 5		
	20	C		Onderzoek de telling. Indien tenslotte alle cijferplaatsen zijn afgewerkt is ook 0 op de tekencijferplaats aangekomen
	21	X 9		} Indien telling nog negatief, herhaal program vanaf 9. Indien telling positief, ga verder met program. Product staat dan in 4.
	22			

enz.

Meestal blijft het product niet binnen de capaciteit van een register, maar zal in de regel dubbele lengte hebben. Ook dan kunnen we een soortgelijk program maken dat alleen iets ingewikkelder wordt.

Het schuiven naar rechts is door het vermenigvuldigen ook overbodig geworden. We kunnen om bijv. een plaats naar rechts te schuiven

ook vermenigvuldigen met $0,1 (= 2^{-1})$ en het meest significante deel van het product behouden.

De logische vermenigvuldiging of collatie wordt gebruikt om een stuk uit een getal te knippen, bijv.:

	01011101100011
gecolleerd met	00000111110000
geeft	00000101100000.

Hetzelfde resultaat kunnen we ook bereiken door eerst 4 plaatsen naar rechts, dan 9 plaatsen naar links en vervolgens 5 plaatsen naar rechts te schuiven. Om alleen de uiteinden te behouden vereist soms nog een keer opbergen of optellen, maar in principe kunnen we het alleen met schuiven af.

De deling kan op een dergelijke wijze als vermenigvuldigen geprogrammeerd worden.

Er zijn nu nog slechts over de opdrachten O, A, S, R, X en C.

Eén van de opbergopdrachten is in ieder geval overbodig. Hebben we alleen S, dan kunnen we altijd met een opdracht O het getal weer terugkrijgen; hebben we alleen R, dan kunnen we door aftrekking het register altijd schoonmaken. We zullen aannemen alleen S te behouden.

Van de optel- en aftrekopdracht is er ook slechts een essentieel. Hebben we alleen de A opdracht, dan kunnen we $a + b$ vormen door te bedenken dat $a + b = -(-a-b)$. Als we dus kunnen aftrekken, dan kunnen we doordat $-.- = +$ ook optellen. Minder evident is het, dat we ook de optelopdracht alleen kunnen behouden. We kunnen namelijk als volgt het complement vormen van een getal

bijv.	01011	
verdubbelen	10110	
nogmaals verdubbelen	01100	
nogmaals verdubbelen	11000	
nogmaals verdubbelen	10000	
Tezamen optellen	10101	is juist $-(01011)$.

Eigenlijk komt dit proces neer op het vermenigvuldigen met $11111 = -1$. Natuurlijk is het geen handig proces. Beter kunnen we alleen de aftrekking behouden.

Nu zijn dus nog over de opdrachten A, S, X en C.

De opdrachten A en S kunnen we combineren tot een enkele opdracht N(nieuwe opdracht), welke betekent: Trek getal uit register n af van de inhoud van de accumulator en berg het resultaat ook in n. Met deze opdracht wordt dus ieder register bij het aflezen ook aangetast. Toch is het mogelijk hiermee nog te programmeren. Als voorbeeld zullen we een program maken om de inhoud van 2 bij de inhoud van 3 op te tellen en het resultaat in register 4 te zetten. 2 en 3 moeten hun getallen ieder behouden. O veronderstellen we schoon bij het begin. Dan luidt het program:

		Stel inhoud van 2 = a, en inhoud van 3 = b.
p	N 4	Trek de inhoud van 4 af, hetwelk bijv. x geeft. Berg x ook op.
p+1	N 4	Trek x van x af en berg 0 op, Accumulator en 4 zijn nu schoon.
p+2	N 2	Neem -a op. Inhoud van 2 = -a.
p+3	N 3	Vorm -a - b
p+4	N 4	Plaats -a - b ook in 4
p+5	N 3	Trek -a - b weer af. Accumulator en 3 schoon
p+6	N 2	Trek -a af. Geeft a in 2 en accumulator
p+7	N 0	Plaats a ook in 0
p+8	N 3	Maak 3 en accumulator schoon
p+9	N 3	
p+10	N 4	Trek -a - b af. Geeft a + b in 4 en accumulator
p+11	N 0	Trek a af. Geeft b in 0 en accumulator
p+12	N 3	Plaats b weer in 3
p+13	N 0	Maak 0 en accumulator weer schoon.

Ook de geconditioneerde opdracht is niet onmisbaar. In de toelichting zullen we voor het gemak normaal gebruik maken van O, A en S in plaats van van N. Alles kan evenwel ook in N geprogrammeerd worden. De geconditioneerde werking valt uiteen in twee handelingen:

1^e het extraheren van het kenmerk waarop getest wordt, 2^e het maken van een variabele opdracht met behulp van het geëxtraheerde kenmerk. Bezien we eerst 2^e. Stel dat we ter beschikking hebben in de accumulator alleen het tekencijfer van een getal en veronderstellen we verder dat de machine 1024 registers heeft en het adres geheel links in een register geplaatst is. Het optellen van het tekencijfer 1 betekent dan het vermeerderen van het adres met 512. We kunnen dan programmeren:

a	O	a+3	Vorm de instructie X a+4 of X a+4+512, afhankelijk van de inhoud van de accumulator
a+1	S	a+2	Plaats deze instructie op a+2
(a+2	)		Ga nu naar a+4 of a+4+512
"a+3	X	a+4"	Constante, nodig om variabele instructie te vormen
a+4			Program gaat hierheen als tekencijfer 0 was. Anders naar a+4+512, waar eventueel een spronginstructie naar een geschiktere plaats kan staan.

Nu het eerste gedeelte nog: het ontdoen van een getal van alle cijfers behalve het tekencijfer. Dit kan als volgt: Veronderstel dat de machine 31 cijferplaatsen heeft in zijn registers. Dan kunnen we eerst 30 plaatsen schuiven naar links. We krijgen dan uitsluitend het meest rechtse cijfer van het te onderzoeken getal op de tekencijferplaats. Met behulp van het in de vorige alinea beschreven stukje program kunnen we een variabele opdracht maken, waardoor het oorspronkelijke getal van zijn meest rechtse cijfer ontdaan wordt. Op dezelfde wijze kunnen we nu door 29 plaatsen te schuiven het getal van het een na rechtse cijfer ontdoen. Tenslotte is alleen nog het tekencijfer over, waarna we dus de eigenlijke "geconditioneerde opdracht" kunnen laten plaatsvinden. Toegegeven wordt dat het proces enigszins omslachtig is, maar niettemin theoretisch mogelijk.

Voor we verder gaan wil ik in dit verband nog wel even het verband tussen de volgende drie opdrachten belichten, nl. de geconditioneerde opdracht, de collatie en het schuiven naar rechts. Heeft men slechts een van deze opdrachten, dan kan men de andere er gemakkelijk uit construeren. Voor de collatie en voor schuiven met behulp van de gecondi-

tionneerde opdracht is dit al toegelicht. Heeft men alleen collatie, dan kan men hiermee het tekencijfer van een getal extraheren, waarna met het beschreven programma'tje de variabele "geconditionneerde" opdracht gemaakt kan worden. Ook de opdracht schuiven naar rechts kan dienen om het tekencijfer te extraheren door 30 plaatsen naar rechts te schuiven. Deze drie opdrachten hebben een element gemeen, dat de optelling slechts rudimentair vertoont: ze kunnen de informatie in een register partiël gebruiken. Er gaat steeds informatie verloren. De geconditionneerde opdracht behoudt alleen het teken, de collatie "knipt" een stuk uit een getal en bij schuiven naar rechts gaat rechts een stuk verloren. De optelling vertoont deze eigenschap slechts in zoverre, dat hierbij de cijfers die de capaciteit overschrijden, verloren gaan. Vandaar dat het alleen met de optelling meer moeite kost dan met behulp van een van de drie genoemde opdrachten.

We hebben nu nog over de opdrachten N en X. De opdracht X kunnen we nog als volgt kwijtraken. We passen nu een dergelijke truc toe als voor de invoer- en uitvoeropdrachten. Een speciaal register bijv. 0 zal zodanig ingericht worden, dat wanneer dit register ingeroepen wordt, de besturing de volgende N opdracht als sprongopdracht zal opvatten. Elke sprongopdracht naar p ziet er dan uit als

	N	O
	N	p

We hebben nu nog maar een essentiële opdracht overgehouden. Dat betekent dus dat we alleen nog maar adressen hebben en dus in feite geen enkele opdracht meer!

Eigenlijk is nu het adres een verkapte opdracht geworden. Er zijn namelijk het X adres 0 en de invoer- en uitvoeradressen die een speciale functie hebben. In ieder geval hoop ik voldoende aangetoond te hebben, dat er niet veel opdrachten zijn, die essentiël genoemd mogen worden.

DYNAMISCHE MAGNETISCHE FLIP-FLOPS

S.J. Duinker

Lezing in het colloquium Moderne Rekenmachines van het Mathematisch Centrum op 29 november 1952 te Amsterdam.

Onder dynamische magnetische flip-flops worden verstaan trigger-schakelingen, o.m. opgebouwd uit magnetische elementen, die om te kunnen functionneren voortdurend gevoed moeten worden door een wisselstroombron. In dit opzicht bestaat dus een essentiël verschil t.o.v. de zg. statische magnetische flip-flops, waarbij gebruik gemaakt wordt van de twee remanentie-toestanden van de hysteresis-kromme van ferromagnetische materialen, nadat deze tevoren door een magnetiserende impuls van voldoende grootte en een bepaalde polariteit bekrachtigd zijn geweest. Deze laatsten werden vroeger reeds behandeld door de heer Loopstra.

De statische flip-flops hebben als voordelen:

- 1e. geen voedingsapparatuur,
- 2e. vasthouden van informatie voor onbepaalde tijd,
- 3e. levensduur vrijwel onbeperkt,
- 4e. klein van afmetingen,

welke eigenschappen deze elementen bijzonder geschikt maken voor toepassing in geheugen.

Als bezwaren kunnen gelden: moeilijkheden bij het uitlezen indien daarbij niet tegelijk de informatie mag verloren gaan en het feit dat de uitleesspanningen voor sommige toepassingen niet geschikt zijn voor de besturing van volgende organen, waardoor toch weer vaak versterking met behulp van electronenbuizen noodzakelijk is.

Tegen de dynamische flip-flops kunnen als bezwaren worden aangevoerd

- 1e. de noodzakelijkheid van een voedingsoscillator,
- 2e. het wegvallen van de informatie bij onderbreking van de voeding,
- 3e. iets grotere afmetingen door groter aantal schakel-elementen.

Hier tegenover staan echter de voordelen van een grote energieversterking, waardoor geen versterkerbuizen nodig zijn en het continu afgeven van een uitgangsspanning, wat het uitlezen vergemakkelijkt en schakeltechnisch veel meer graden van vrijheid oplevert. Bovendien zijn dynamische flip-flops in hun werking bepaald door een werkelijke instabiliteit, wat ze minder afhankelijk maakt van de vorm en hoogte der hun besturende impulsen.

Van de dynamische flip-flops zullen twee typen worden behandeld, waarbij de voor de trigger-werking noodzakelijke instabiliteiten achtereenvolgens veroorzaakt worden door positieve terugkoppeling van een magnetische versterker en door zg. ferro-resonantie.

I. Instabiele magnetische versterker

Eerst zal in het kort het principe van magnetische versterking worden toegelicht. Magnetische versterkers hebben als essentiële componenten ferro-magnetische kernen voorzien van een of meer wikkelingen, waarbij door een simultane gelijk- en wisselstroom magnetisatie een doelbewust gebruik gemaakt wordt van de niet-lineariteit van de B-H-karakteristiek.

In principe is de werking als volgt (fig. 1):

Een niet-lineaire transformator wordt aan primaire zijde gevoed door een gelijkstroombron I , terwijl de secundaire winding is opgenomen in een wisselstroomcircuit bestaande uit een belastingsimpedantie Z en een voedingsoscillator V .

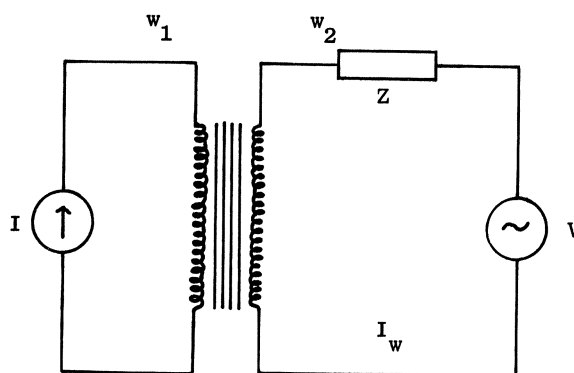


fig. 1

Bij constante amplitude en frequentie V hangt de effectieve waarde van de wisselstroom - $\text{aw } I_w w_2$ af van de gelijkstroomvoormagnetisatie $I w_1$ zoals kwalitatief is aangegeven door kromme a in fig. 2.

Voor het sturen van I door

w_1 is een zeker vermogen P_1 nodig. In Z wordt door I_w een vermogen P_2 afgestaan. Indien $P_2 > P_1$ kan gesproken worden van een energieversterking. Een magnetische versterker zet dus een gelijkstroomvermogen om in een groter wisselstroomvermogen, of, na gelijkrichting, in een groter gelijkstroomvermogen.

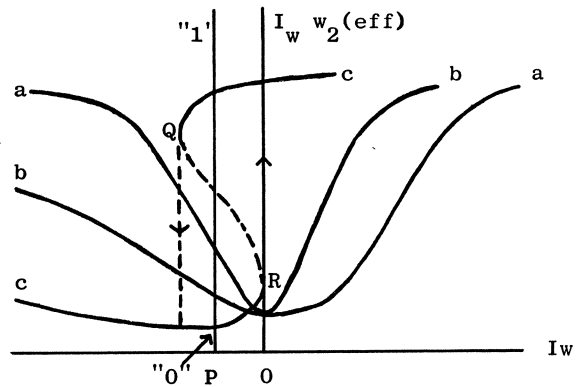


fig. 2

Indien de voedingsfrequentie maar hoog genoeg gemaakt kan worden, kunnen ook wisselstromen versterkt worden, waarbij men uiteraard te maken krijgt met modulatie. Om het oorspronkelijke signaal terug te verkrijgen moet gedemoduleerd worden, waardoor men noodzakelijkerwijs de hoogste signaalfrequentie een factor 10 à 15 lager moet kiezen dan de voedingsfrequentie.

Door terugkoppeling van de gelijkgerichte belastingsstroom $\bar{I}_w$ naar een extra ingangswikkeling w_3 (fig. 3) kan de versterkingsfactor aanzienlijk worden opgevoerd (kromme b in fig. 2). Maakt men w_3 groot genoeg dan kan een kromme c verkregen worden, die een gedeelte met negatieve helling vertoont, dat instabiel is. Bij een polarisatie P zijn twee evenwichtsstanden mogelijk: "0" en "1", resp. corresponderende met een lage of hoge waarde van de wisselstroom.

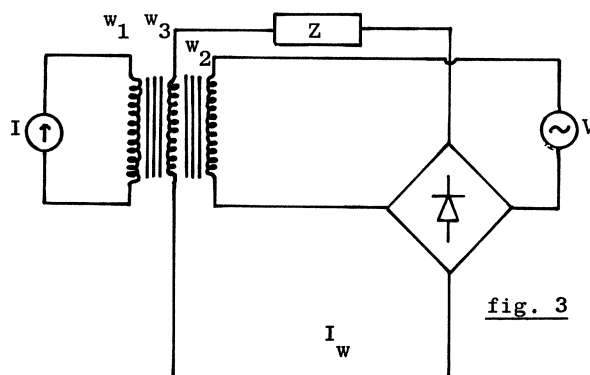


fig. 3

Deze eigenschap is bruikbaar voor een trigger. Vanuit "0" zal men onder invloed van een positieve impuls via R discontinu overgaan op de bovenste tak en tenslotte in "1" belanden. Verdere positieve impulsen hebben op de toestand "1" geen invloed meer, doch een negatieve impuls brengt het circuit via Q naar "0" terug. Men heeft hier dus te maken met een werkelijk instabiel verschijnsel, waarbij de impulsbreedte en flanksteilheid niet essentiële zijn voor een transitie, zulks in tegenstelling met statische flip-flops, waar het een volkomen stabiel verschijnsel betreft en de uitgangsspanningen indirect in relatie staan met de vorm van de impuls.

Bovenstaand principe is door J.G. Miles (Rep.: Saturable Core Reactors as Digital Computer Elements; Navy Dept. Bureau Ships; Contr. No b s r 42001, 17/6/'49) toegepast in progressive- en gated-carry counters en registers, waarvan enkele voorbeelden zullen worden behandeld.

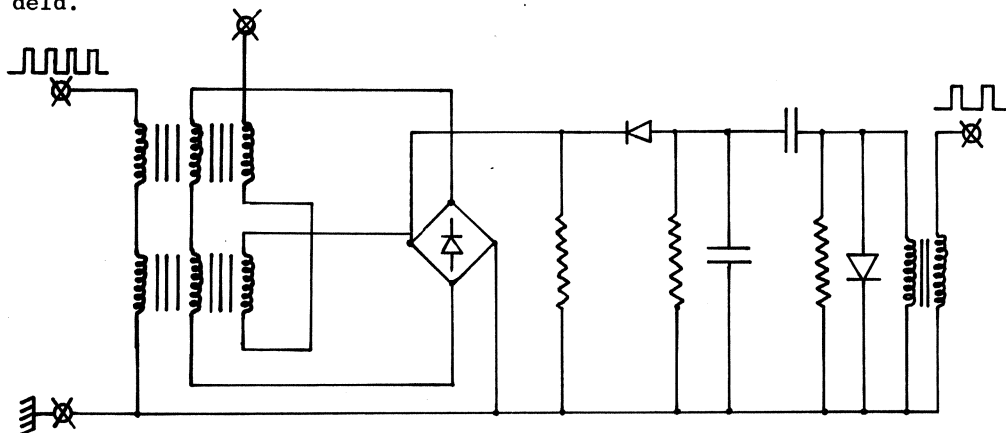


fig. 4

Progressive Counter

II. Ferro-resonantie flip-flops (C. Isborn, Electronics 25, 4 (1952), 121-123).

Deze berusten in hun werking op het feit, dat in een serie circuit bestaande uit een niet-lineaire zelfinductie en een al of niet lineaire capaciteit onder invloed van een wisselspanningsbron instabiliteiten in de stroomtoestand kunnen optreden. Bij geschikte dimensionering ziet het verband tussen bv. de spanning en wisselstroom, bij geleide-

lijke variatie er uit als in fig. 5 is geschetst.

Bezwaarlijk is echter dat de amplitude van de generatorspanning gevarieerd moet worden om transitie's te verkrijgen. Deze moeilijkheid kan op betrekkelijk eenvoudige wijze worden opgelost door twee, overigens gelijke circuits parallel via

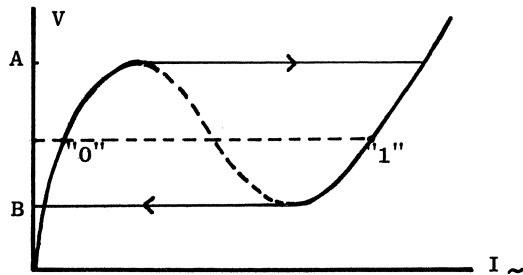


fig. 5

een serie-reactantie Z aan te sluiten op de generator (fig. 6).

Men dimensioneert het circuit zodanig dat indien beide circuits in de hoge stroomtoestand ("1") zouden verkeren, de spanningsval over Z zo groot is, dat de spanning V' over het parallelcircuit beneden de waarde B zou komen en anderszins, dat, indien beide circuits in de "0"-toestand zouden verkeren, dat V' boven de waarde A zou uitstijgen.

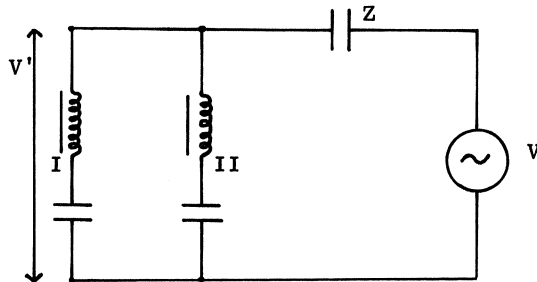


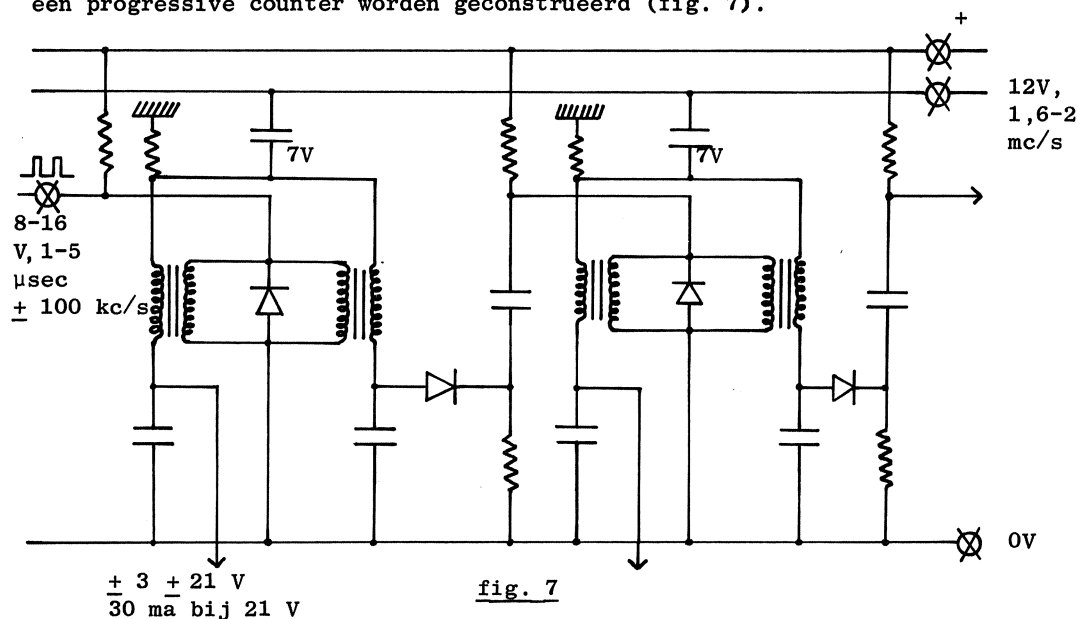
fig. 6

Op deze wijze is te bereiken, dat alleen als het ene circuit in "0" en het andere in "1" verkeert, V' in de gewenste marge $A-B$ komt te liggen. De toestand van elke flip-flop kan worden afgelezen aan de spanning over de zelfinductie op de capaciteit. De amplitude van de wisselspanning over deze elementen is een aanduiding van de inhoud. Eventueel kunnen deze spanningen nog gelijkgericht worden. Door de niet-lineaire resonantie is de spanning over de elementen een factor 3 à 6 groter dan V' . De uitgangen van de flip-flop zijn door het versterkende karakter van de schakeling vrij zwaar belastbaar, b.v. met 0,5 watt. Voorziet men beide kernen nog van een ingangswikkeling, dan kan men de flip-flop van stand laten veranderen door toevoeren van een impuls aan de flip-flop met de lage stroomtoestand. Hierdoor wordt n.l. de effectieve zelf-

inductie van deze tak verlaagd, waardoor deze een grotere stroom gaat voeren en begint over te gaan naar de hoge stroomtoestand. Tegelijkertijd neemt echter de spanningsval over Z toe, dus wordt V' lager, waardoor de tweede flip-flop gedwongen wordt naar de lage stroomtoestand. Dit proces speelt zich in een aantal perioden van de voedingsspanning ($0,5 - 2 \text{ mc/s}$) af.

Deze schakeling is vergelijkbaar met een dubbelbuis flip-flop; zij bezit 2 inputs en 2 outputs, welke laatsten in tegenfasen werken. Door parallelschakeling van de twee inputwikkelingen kan bereikt worden, dat de flip-flop van stand verandert onder invloed van gelijkgerichte impulsen.

Door gelijkrichting en differentiatie van het uitgangssignaal kan een progressive counter worden geconstrueerd (fig. 7).



Enkele algemene gezichtspunten van ferro-resonantie flip-flops.

Voorlopig nog vrij kostbaar, ca. 20 \$ per stuk. Deze hoge prijs hoofdzakelijk te wijten aan dure en moeilijk te bewikkelen kernen ($1/8 \text{ mil}$; $4/79 \text{ Mo-permalloy}$). Toepassing van ferrietkernen kan leiden tot hogere frequenties en tot lagere prijs.

Kleine afmetingen ($< 0,1$ cubic in): b.v. decade-eenheid, compleet met neon-uitleesbuisje $\pm 1'' \times 2'' \times 3''$.

Betrouwbaarheid, geen gloeidraden; stabiel, geen verloop van de karakteristiek; gemakkelijker circuit-opbouw; weinig warmte-ontwikkeling; betrekkelijk gering energieverbruik (per decade ± 2 watt).

EEN OF MEER ADRESCODES IN REKENMACHINES?

C.S. Scholten

Lezing in het colloquium Moderne Rekenmachines van het Mathematisch Centrum op 20 december 1952 te Amsterdam.

De bedoeling van deze voordracht is, een beknopt overzicht te geven van de mogelijke codes en waar mogelijk, aan de hand van een voorbeeld de voor- en nadelen van de verschillende mogelijkheden tegen elkaar af te wegen. Vooropgesteld zij, dat het geenszins de bedoeling is in dezen naar volledigheid te streven, al was het alleen maar wegens het feit dat het probleem van het "gemiddelde programma" zelfs nog niet bij benadering is opgelost, zo het al ooit serieus is aangepakt.

Om de gedachten te bepalen zullen we ons voorlopig voorstellen dat we te doen hebben met een seriemachine met een of andere vorm van cyclisch geheugen (b.v. een magnetische trommel), waaruit we onze woorden eveneens in serie zullen uitlezen.

Nemen we aan, dat we in een track 1024 bits kunnen bergen en dat een volledig woord uit 32 bits bestaat, dan kunnen we dus 32 hele woorden op een track bergen. Deze zullen we gewoon achter elkaar op de omtrek neerzetten, zodat, wanneer we afzien van een gap, elk woord $\frac{1}{32}$ van de omtrek beslaat.

Het is duidelijk dat de gemiddelde aanhaaltijd voor een woord (gerekend tot en met het binnenlopen van de laatste digit) ongeveer een halve omwenteling (cyclus) bedraagt. Iets nauwkeuriger bedraagt deze aanhaaltijd als we even $\sqrt{N}$ woorden per track aannemen i.p.v. 32 $\tau_{1N} = \frac{1}{2} + \frac{1}{N}$ (in omwentelingen).

De tweede term verdisconteert het feit, dat de woorden een van nul verschillende fractie van de omtrek in beslag nemen.

Willen we meer woorden tegelijk aanhalen, dan wordt de aanhaaltijd langer, de aanhaaltijd per woord echter korter. Willen we b.v. twee woorden aanhalen, onafhankelijk van hun volgorde, dan is de aanhaaltijd $\tau_{2N} = \frac{2}{3} + \frac{7}{6N}$, waarbij tevens gemiddeld is over de mogelijke posities van de twee woorden op de omtrek. De aanhaaltijd per woord is dus $\frac{1}{3} + \frac{7}{12N}$.

Volledigheidshalve vermelden we nog dat onder dezelfde voorwaarden geldt

$$\tau_{mN} = \frac{m}{m+1} + \frac{3m+1}{2(m+1)N} \quad (m \leq N)$$

en de aanhaaltijd per woord dus

$$\tau_{\frac{mN}{m}} = \frac{1}{m+1} + \frac{3m+1}{2m(m+1)N}$$

We onderscheiden nu de volgende codes:

1. 1-adrescode. Bij dit systeem bevat de opdracht behalve het functiegedeelte (f) slechts een adres (a). Afgezien van enkele speciale opdrachten is de gang van zaken zo, dat de arithmetische bewerking, aangegeven in het functiegedeelte wordt uitgevoerd op het getal op het aangegeven adres en op dat getal dat zich op een vast adres, zeg adres 0, bevindt. Het resultaat van de bewerking kunnen we naar verkiezing weer op adres a of op adres 0 terugschrijven. Kiezen we de laatste mogelijkheid en bedenken we dat we adres 0 tweemaal moeten passeren, hetgeen ons minstens een omwenteling kost en dat we bovendien een mogelijkheid moeten hebben, het gevonden resultaat te bewaren tot we weer op adres 0 zijn, waar we het resultaat weer kwijt kunnen, dan ligt de bouw van een accumulator zeer voor de hand. Deze fungeert dan als snelle geheugenplaats. De opdracht f/a heeft dan betrekking op het getal op adres a en het getal in de accumulator (A), terwijl het resultaat weer in A wordt afgeleverd. (Van technische complicaties bij vermenigvuldiging en deling wordt hier afgezien.)

Gezien het feit, dat er geen extra adres gebruikt wordt om de plaats van de volgende instructie aan te geven, volgen we in dit geval de conventie, dat we de opdrachten in die volgorde afhandelen, waarin ze in het geheugen zijn neergezet.

2. 1+1 - adrescode. Het tweede adres wordt hier gebruikt om aan te geven op welke plaats in het geheugen de volgende opdracht gehaald moet worden.

Overigens als boven.

3. 2-adrescode. Hierbij wordt op beide aangegeven adressen gelezen en/of geschreven. Op het eerste gezicht lijkt het aangewezen b.v. de optelopdracht a 0 b aldus te interpreteren: tel de getallen op de adressen a en b op. Rest nog slechts de vraag, wat we met het resultaat zullen doen. Opbergen in A heeft weinig zin, tenzij we A ook werkelijk als geheugenplaats beschouwen en een adres geven. Anders kunnen we n.l. niet meer op de som opereren, tenzij we deze eerst met een andere opdracht in het geheugen terugschrijven.

Een andere interpretatie is: tel (a) op bij A en berg het resultaat in b. Dit heeft het nadeel, dat we in vele gevallen niet de bedoeling hebben het resultaat in het geheugen terug te schrijven, doch er eerst nog andere bewerkingen op willen uitvoeren.

4. 2+1 -adrescode. Zie verschil tussen 1- en 1+1 -adrescode.

5. 3-adrescode. De meest voor de hand liggende interpretatie van de opdracht $f/_{abc}$ luidt nu: Voer de operatie f uit op (a) en (b) en schrijf het resultaat op c. Alle drie adressen zijn nu variabel en het nut van een accumulator is nu slechts gelegen in het snel bereikbaar zijn als geheugenplaats.

6. 3+1 -adrescode. Duidelijk na het voorgaande.

We zullen nu de verdiensten van deze verschillende codes even nagaan aan de hand van een subroutine voor worteltrekking. Precieser gezegd: De subroutine vormt uit de (positieve) breuk α de opvolgende benaderingen $\alpha_0 \alpha_1 \dots$ voor $\sqrt{\alpha}$ volgens $\alpha_{j+1} = \frac{1}{2}(\alpha_j + \frac{\alpha}{\alpha_j})$ en stopt wanneer $\alpha_j - \alpha_{j+1} < \delta$ is geworden. Als beginschatting nemen we de j "bijna 1".

In de een-adrescode zoals b.v. bij de ARRA in gebruik, zou deze routine als volgt luiden. We komen de routine binnen met het adres c waar we vandaan gekomen zijn in A en α b.v. in S.

Aanroep:	c	0/c	
	c+1	7/d	
	c+2	...	
Subroutine:	d+0	0	) ⁷ /2(
	1	10	d+17
	2	11	d+18
	3	0	)<1(
d+15 →	4	10	d+19 1
	5	0	d+18 1
	6	13	d+19 2
	7	3	d+20 1
	8	8	d+19 1
	9	14	1 1
	10	0	d+21 1
	11	7	d+15 0 →
	12	8	d+21 1

	13	0	d+19	1
	14	7	d+4	1 ==>
d+11 →	15	8	d+21	
	16	0	d+19	
	17	(7	c+2)	
	18	(α)		
	19	(α_j)		
	20	v		
	21	δ		

De hierin gebruikte opdrachten hebben de volgende betekenis:

- 0 a Vormt (a)+(A) en zet dit in A
- 3 a Zet (S) in A en a.
- 7 a Verplaatst besturing naar a als (A) $\geq$ 0 is. Neemt anders volgende opdracht.
- 8 a Vervangt (A) door (A)-(a).
- 10 a Zet (A) in a en maakt A schoon.
- 11 a Zet (S) in a en maakt A schoon.
- 13 a Deelt (A) afgerond door (a) en plaatst quotient in S. Maakt A niet schoon.
- 14 a Schuift (A) a plaatsen naar rechts. Links wordt het teken van (A) aangevuld.

Achter elke opdracht van de cyclus in deze subroutine is aangegeven hoeveel omwentelingen deze opdracht kost. Hierbij is aangenomen dat de deling ongeveer een hele omwenteling kost, hetgeen in overeenstemming is met onze afspraak 32 woorden van 32 digits op de omtrek te plaatsen.

We komen zodoende tot 11 omwentelingen per cyclus. Men ziet dat het feit, dat deling (en vermenigvuldiging) c.a. 32 woordlengten in beslag nemen nog slechts 10% in tijd uitmaakt. Nemen we in plaats van adres d+20 een algemeen "vuilnis" adres op een andere track, dan moeten we nog de schakeltijd in rekening brengen. Nemen we aan dat we een magnetische trommel hebben met 20 msec omwentelingstijd, dan kunnen we nog met relais schakelen met een operatietijd, die we conservatief op 10 msec zullen stellen. Dit scheelt in dit geval 1 omwenteling. Hebben we daarentegen een megahertz kwiktank met een cyclus van 1 msec dan is dit uiteraard ontoelaatbaar. De schakeltijd zal dan van de orde van 500 μ sec of minder moeten zijn.

De geheugenruimte die dit programma inneemt is vrij kritisch op het totaal aantal geheugenplaatsen. Nemen we aan dat een opdracht een halve woordlengte beslaat (b.v. 4 functiedigits en 11 of 12 adresdigits), dan komen we in totaal tot 13 woorden van 32 digits. Hiervan neemt de cyclus met bijbehorende getaladressen er $9\frac{1}{2}$ voor zijn rekening.

Laat ons zien wat er van dit alles terecht komt in systeem no. 2, dat ons demogelijkheid van optimaal programmeren biedt, d.w.z. de mogelijkheid onze opdrachten en getallen zodanig in het geheugen te plaatsen dat zodra we een opdracht hebben ingelezen, juist het adres waar de opdracht betrekking op heeft, gelezen kan worden en omgekeerd, zodra een opdracht geheel is uitgevoerd, juist de volgende opdracht gelezen kan worden.

We kunnen al dadelijk zeggen, dat het minimum aantal omwentelingen per cyclus 5 zal zijn zolang we het getal α_j slechts op een plaats op de omtrek neerzetten, aangezien deze plaats 4 x wordt aangeroepen en we voor de deling ook nog een omwenteling nodig hebben. Dit minimum is inderdaad bereikbaar mits we de opdrachten d+12 en d+13 uit ons vorige programma laten stuivertje wisselen, (adres van δ wordt ook 2 x aangeroepen!). De cyclus zou er dan b.v. als volgt uitzien (beginnende op plaats 0). De (relatieve) verwijzingsadressen staan tussen haakjes.

0	0	1(+2)
1	α	
2	13	3(+2)
3	α_j	
4	3	5(+2)
5	v	
6	8	3(+1)
7	14	1(+2)
8	8	10(+5)
9	9	10(+2)
10	δ	
11	7	15(+1) →
12	0	3(+28)
13	10	3(+1)
14	7	0 ==>

De benodigde geheugenruimte is nu echter gestegen tot 15 hele woorden voor de cyclus.

Zijn we bereid 5 geheugenplaatsen extra ter beschikking te stellen dan kunnen we het aantal omwentelingen nog drukken tot 3, als volgt:

0	0	1(+2)
1	α	
2	13	3(+2)
3	α_j	
4	3	5(+2)
5	v	
6	8	8(+3)
7	2	8(+3)
8	α_j	
9	14	1(+2)
10	10	17(+9)
11	0	12(+2)
12	δ	
13	7	20(+1) $\rightarrow$
14	8	15(+2)
15	δ	
16	0	17(+2)
17	α_j	
18	2	3(+21)
19	7	0 $\Rightarrow$

Willen we deze subroutine nu ook in een 2-adrescode gaan bekijken, dan moeten we ons eerst even realiseren hoe de opdrachten er ongeveer zullen gaan uitzien. Uitgaande van de opmerking die dienaangaande reeds is gemaakt, zullen we veronderstellen een accumulator (A) en nog een hulpregister (S) te bezitten, beiden als geheugenplaats aanroepbaar.

We stellen nu de volgende opdrachten voor:

0	a	b	$(a) + (b) \rightarrow A$
1	a	b	$(a) - (b) \rightarrow A$
2	a	b	$\frac{(a)}{(b)} \rightarrow S$ (afgerond)
3	a	b	(A) a plaatsen naar rechts en schrijf het resultaat op b en in A.
4	a	b	$(a) \rightarrow b$
5	a	b	Vorm $(a) + (A)$ en indien dit ≥ 0 is, verplaats dan de besturing naar b. Anders gewoon volgende opdracht

De cyclus uit de subroutine zou nu de volgende vorm krijgen:

0	2	7	8	2.
1	1	S	8	1
2	3	1	S	1
3	5	9	10	1 →
4	0	8	S	1
5	4	A	8	1
6	5	S	0	1 ==>
7	α			
8	α_j			
9	δ			

Tijd: 8 omwentelingen.

Geheugenruimte: 10 hele woorden.

De optimaal geprogrammeerde 2-adrescode met of zonder ter beschikkingstellen van extra geheugenruimte, biedt weinig interessante gezichtspunten meer, daar de overwegingen die ons bij de 1+1 adrescode tot een minimum van 5 resp 3 omwentelingen brachten, ook nu gelden. Wel moeten we opmerken dat we langzamerhand met een ruimteprobleem te kampen krijgen. Bij 2048 woorden en 16 opdrachten blijven er nog 6 digits over voor ons verwijzingsadres, dat we nu wel relatief moeten nemen.

Volledigheidshalve geven we de programma's nog even:

0	2	1	2	(+3)
1	α			
2	α_j			
3	1	S	2	(+1)
4	3	1	S	(+2)
5	5	S	0	==>
6	5	7	10	(+2) →
7	δ			
8	0	2	S	(+1)
9	4	A	2	(+28)

Tijd: 5 omwentelingen.

Geheugenruimte: 10 hele woorden.

0	2	1	2	(+3)
1	α			

2	α_j			
3	1	S	5	(+3)
4	4	A	5	(+3)
5		α_j		
6	3	1	S	(+2)
7	4	A	11	(+6)
8	5	9	14	(+2) →
9	6			
10	0	11	S	(+2)
11		α_j		
12	4	A	2	(+24)
13	5	S	0	==>
14				

Tijd: 3 omwentelingen

Geheugenruimte: 14 hele woorden.

De 3-adrescode zonder accumulator blijkt in dit geval geen nuttig effect te sorteren. Willen we onze opdrachten niet al te veel ad hoc kiezen dan zouden we b.v. kunnen hebben:

0	a b c	$(a)+(b) \rightarrow c$
1	a b c	$(a)-(b) \rightarrow c$
2	a b c	$(a) \times (b) \rightarrow c$ (afgerond)
3	a b c	$(a)/(b) \rightarrow c$ (afgerond)
4	a b c	$2^b(a) \rightarrow c$
5	a b c	$2^{-b}(a) \rightarrow c$
6	a b c	$(a)+(b) \geq 0$ dan besturing naar c.

Weliswaar bevat onderstaand programma slechts 6 opdrachten, doch daar staat tegenover dat we nu in geen geval onze woorden van 32 binalen zullen kunnen handhaven. Houden we vast aan een geheugencapaciteit van 2048 plaatsen en de mogelijkheid van 16 verschillende opdrachten, dan komen we tot een minimum van 37 binalen.

0	3	6	7	8
1	1	8	7	9
2	5	9	1	10
3	6	10	11	12 →
4	0	7	10	7

```

5   6   7   10   0 ==>
6   α
7   αj
8
9
10
11  δ

```

Tijd: 8 omwentelingen

Er is geen sprake van dat deze routine zodanig geprogrammeerd kan worden, dat hij ook afgewerkt kan worden in drie omwentelingen. Daarvoor is het aantal adressen dat twee- of meermalen aangeroepen wordt te groot. Hierbij komt nog dat het invoeren van nog een adres weer een uitbreiding van het aantal binalen vereist. Natuurlijk is het mogelijk door wederinvoering van A onze opdrachten nog wat werkzamer te maken, door twee operaties op drie getallen uit te voeren en het resultaat in A op te bergen (dus b.v. de opdracht $(a) \times (b) + (c) \rightarrow A$). Wellicht is het dan mogelijk ook in deze code optimaal te programmeren, doch veel schieten we hier niet mee op. Van de tot nu toe verkregen resultaten geven we nog even een overzicht. De gegevens slaan op de cyclus in de subroutine.

	Tijdsduur (in omw.)	Geheugenruimte	Opmerkingen
1 adres	11	9½	
1+1	5	15	
	3	20	
2	8	10	
2+1	5	10	Slechts 6 digits
	3	14	voor verwijzingsadres
3	8	12	minimaal 37 digits
			per woord

We merken op dat bij de optimaal geprogrammeerde gevallen van 3 omwentelingen de deling een aanzienlijke rol gaat spelen in de tijdsduur.

Nu is het inderdaad mogelijk de tijdsduur van de deling (en vermenigvuldiging) te bekorten. Wanneer we de beide getallen die we willen delen eenmaal hebben ingelezen, kunnen we deling op een b.v. 10 x hogere frequentie uitvoeren. Dit heeft echter het nadeel dat de optelapparatuur 10 x sneller moet kunnen werken dan voor een optelopdracht vereist is.

Een andere methode is de volgende:

We splitsen onze 32 binalen in 8 groepen van 4 digits en behandelen onze getallen verder alsof ze in het 16-tallig stelsel waren neergeschreven. Een woord schrijven we nu op 4 tracks en wel zo, dat we telkens de 4 binalen van 1 sedecimaal tegelijk inlezen. Dit vereist natuurlijk 4 x zoveel lees- en schrijffapparatuur als vroeger. In een omwenteling kunnen we dan 128 woorden lezen i.p.v. 32. Leren we onze machine de tafels van vermenigvuldiging van $0 \times 0 = 0$ tot en met $15 \times 15 = 1$ met een carry van 14, dan behoeft een vermenigvuldiging slechts 8 woordlengten te duren.

In woordlengten is de winst een factor 4, in tijd zelfs een factor 16. Mede in verband met het feit dat we nu de beschikking hebben over 128 woorden per omtrek, kost het nu niet de geringste moeite de gehele subroutine in twee omwentelingen te programmeren in een van de optimaal programmeerbare codes.

Willen we de subroutine nog sneller maken, dan kunnen we twee dingen doen:

a. De cyclus uit de subroutine meermalen langs de omtrek uitschrijven.

In het laatste beschouwde geval zouden we dit b.v. 4 maal kunnen doen. Dit helpt echter niet zo bar veel. Elke keer dat ik een nieuwe benadering α_j heb gevonden, zal ik deze ergens op de omtrek neer moeten zetten.

Tijdens de volgende cyclus wil ik die waarde weer gebruiken. Hoe slim ik de opdrachten ook neerzet, dit kost minstens een omwenteling. I.h.a. bedraagt in dit geval de tijd voor 1 cyclus bij $m \times$ uitschrijven $\frac{m+1}{m}$ omwentelingen.
b. Een of meer snelle geheugenplaatsen invoeren. Hebben we 1 snelle geheugenplaats, waar we onze α_j in opbergen, dan kunnen we b.v. in de 1+1 adrescode zonder meer 1 omwenteling halen en bovendien nog geheugenruimte sparen, daar de extra schrijfoopdrachten en opbergplaatsen van de α_j vervallen. Het programma zou er dan b.v. als volgt uitzien: (de snelle geheugenplaats is met G aangegeven).

0	0	1	(+2)
1	α		
2	13	G	(+8)
...			
10	3	11	(+2)
11	v		
12	8	G	(+1)

13	14	1	(+2)
14	...		
15	0	16	
16	δ		
17	7	...	(+1) →
18	0	19	
19	δ		
20	0	G	
21	10	G	
22	7	0	==>

Wanneer we nu nog eens de methode genoemd onder a toepassen, blijkt deze veel meer effect te sorteren. Bij m keer uitschrijven blijkt een cyclus slechts $\frac{1}{m}$ omwenteling te kosten.

Van de andere mogelijke organisatievormen van onze machine wil ik nog slechts een geval even aanstippen, en wel een extreem. Gesteld dat we een programma hebben zonder besturingsverplaatsingen. We lezen nu onze opdrachten parallel van de trommel (d.w.z. alle 32 digits tegelijk), nemen voor al onze getallen snelle geheugenplaatsen, maken voorts een instantane opteller en vermenigvuldiger (deler) en zijn nu in staat onze opdrachten met klokpuls-frequentie uit te voeren, d.w.z. 20μ sec per opdracht.

Geen slecht resultaat voor een magnetische trommel! Over de kosten van een instantane vermenigvuldiger zullen we het echter maar niet hebben, evenmin als over het voorkomen van programma's zonder besturingsverplaatsingen.

Zoals reeds in de inleiding werd opgemerkt, is het voorgaande geen uitputtend onderzoek geweest. Verre van dien. De bedoeling was slechts te onderzoeken wat we, bij niet al te onredelijke programma's, van de verschillende organisatievormen mogen verwachten. Tevens moge duidelijk geworden zijn, dat een en ander sterk afhangt van de aard van het programma, i.h.b. van het aantal vermenigvuldigingen en delingen en, bij cyclische geheugen, van het aantal cycli in het programma en hun gemiddelde lengte.

DE MARK IV REKENMACHINE VAN HARVARD UNIVERSITY

G.A. Blaauw

Lezing in het colloquium Moderne Rekenmachines van het Mathematisch Centrum op 31 januari 1953 te Amsterdam.

De Mark IV rekenmachine is in de jaren 1950-1952 door de leden van het Computation Laboratory van Harvard University ontworpen en geconstrueerd en is momenteel in een ver gevorderd test-stadium. De machine is gebouwd voor het Flight Research Laboratory van de United States Air Force en geplaatst in het Computation Laboratory. De bediening van de rekenmachine zal geschieden door de leden van het Computation Laboratory.

Mark IV is de vierde rekenmachine, welke onder leiding van Prof. Howard Aiken gebouwd is. De algemene organisatie zowel als vele constructie details zijn van hem afkomstig.

Geheugen

De vier belangrijkste delen waaruit Mark IV bestaat zijn: a) het geheugen, b) het arithmetische gedeelte, c) de in- en uitvoer-organen, d) het opdracht gedeelte. De verbindingen tussen deze delen zijn in het bijgaande blokschema aangegeven. Een getal kan in de richting van de pijlen worden gestuurd. De machine is in twee gedeelten onderverdeeld het β gedeelte en het γ gedeelte, die door de zgn. "Fast In Bus" en "Fast Out Bus" zijn verbonden. De gedeelten aan de β zijde van de machine lezen getallen via de "Fast In Bus" en schrijven getallen via de "Fast Out Bus". De gedeelten aan de γ zijde van de machine lezen via de "Fast Out Bus" en schrijven via de "Fast In Bus". Getallen blijven bewaard in de eenheden, waaruit zij gelezen zijn tot zij vernietigd worden bij het schrijven van een nieuw getal. Een getal kan

alleen verplaatst worden van een β eenheid naar een γ eenheid en omgekeerd.

Het geheugen van de machine kan in drie gedeelten worden verdeeld:

a) Snel geheugen, b) langzaam geheugen en c) hulp registers.

Het snelle geheugen bestaat uit 210 registers, die elk in staat zijn een decimaal getal van 16 cijfers met bijbehorend teken te bewaren. De cijfers worden door een gecodeerd decimaal getalstelsel van vier binaire cijfers weergegeven. Getallen worden in serie uit de registers gelezen. De tijd nodig voor het lezen van een getal vormt de machinecyclus en wordt getaltijd genoemd. Dit is 1,2 milliseconde. De tijd nodig voor het lezen van een cijfer, een zgn. cijfertijd, bedraagt 60 microseconden. Er zijn twintig cijfertijden in een getaltijd. Alle registers van het snelle geheugen zijn β eenheden.

Het langzame geheugen heeft plaats voor 4000 getallen op het oppervlak van een magnetische trommel. De trommel heeft een grote accessietijd, daar relaisschakeling gebruikt wordt. Deze wordt gedeeltelijk gecompenseerd door het schrijven of lezen in groepen van tien getallen. Het trommelgeheugen behoort tot de β zijde van de machine. Het is met het snelle geheugen verbonden door middel van 20 overdrachtregisters. Deze overdrachtregisters kunnen ook als snelle geheugenregisters gebruikt worden.

De hulpregisters worden onderscheiden in:

- a) γ overdrachtsregister dat als tijdelijk geheugen dient, wanneer een getal van een β register naar een ander β register verplaatst wordt. Dit register wordt ook gebruikt om getallen van vier cijfers via de opdracht in de machine te brengen.
- b) β overdrachtsregister, dat als tijdelijk geheugen dient, wanneer een getal van een γ register naar een ander γ register verplaatst wordt. Dit register wordt ook gebruikt als schuivend register.
- c) Schuivend controleregister. Bepaalt het aantal cijferplaatsen, dat een getal verschoven uit het schuivend register gelezen moet worden.

Het getal in het schuivend register blijft onveranderd door deze lees-operatie. Schuivend register en controleregister worden ook gebruikt voor het normaliseren van getallen.

- d) Sprong-register. Hierin wordt het adres bewaard, waarheen de laatste sprong gemaakt is.
- e) Adresregister. Hierin is het adres van de opdracht, die wordt uitgevoerd, beschikbaar.
- f) I-J-register. Gebruikt voor de manipulatie van opdrachten.
- g) K-register. Gebruikt voor de manipulatie van opdrachten, die betrekking hebben op de band-mechanismen.
- h) Conditie-register. Bepaalt het gehoorzamen van een conditionele sprong.
- i) Langzaam geheugen controle-register. Bepaalt het adres op de magnetische trommel, waar gelezen of geschreven zal worden.

Deze hulpregisters, uitgezonderd het γ overdrachtregister, behoren tot de β zijde van de machine.

Arithmetisch gedeelte

De volgende arithmetische apparatuur is beschikbaar:

- a) Twee accumulatoren
- b) Vermenigvuldiger
- c) Deler.

Alle arithmetische apparatuur is aan de γ zijde van de machine geplaatst.

Elke accumulator kan een getal bij de voorgaande som optellen in een getaltijd. Een getal kan schoon of accumulatief in de opteller geschreven worden. Speciale opdrachten zijn beschikbaar om getallen van $n \times 16$ cijfers op te tellen.

De vermenigvuldiger kan twee getallen vermenigvuldigen gedurende de 7 getaltijden, die op het lezen van de vermenigvuldiger volgen.

De machine is beschikbaar voor andere bewerkingen gedurende de 7 getaltijden van de vermenigvuldiging.

De deler kan twee getallen op elkaar delen gedurende de 20 getaltijden, die volgen op het lezen van de deler. Dit geschiedt wederom onafhankelijk van de rest van de machine.

In- en uitvoer-organen

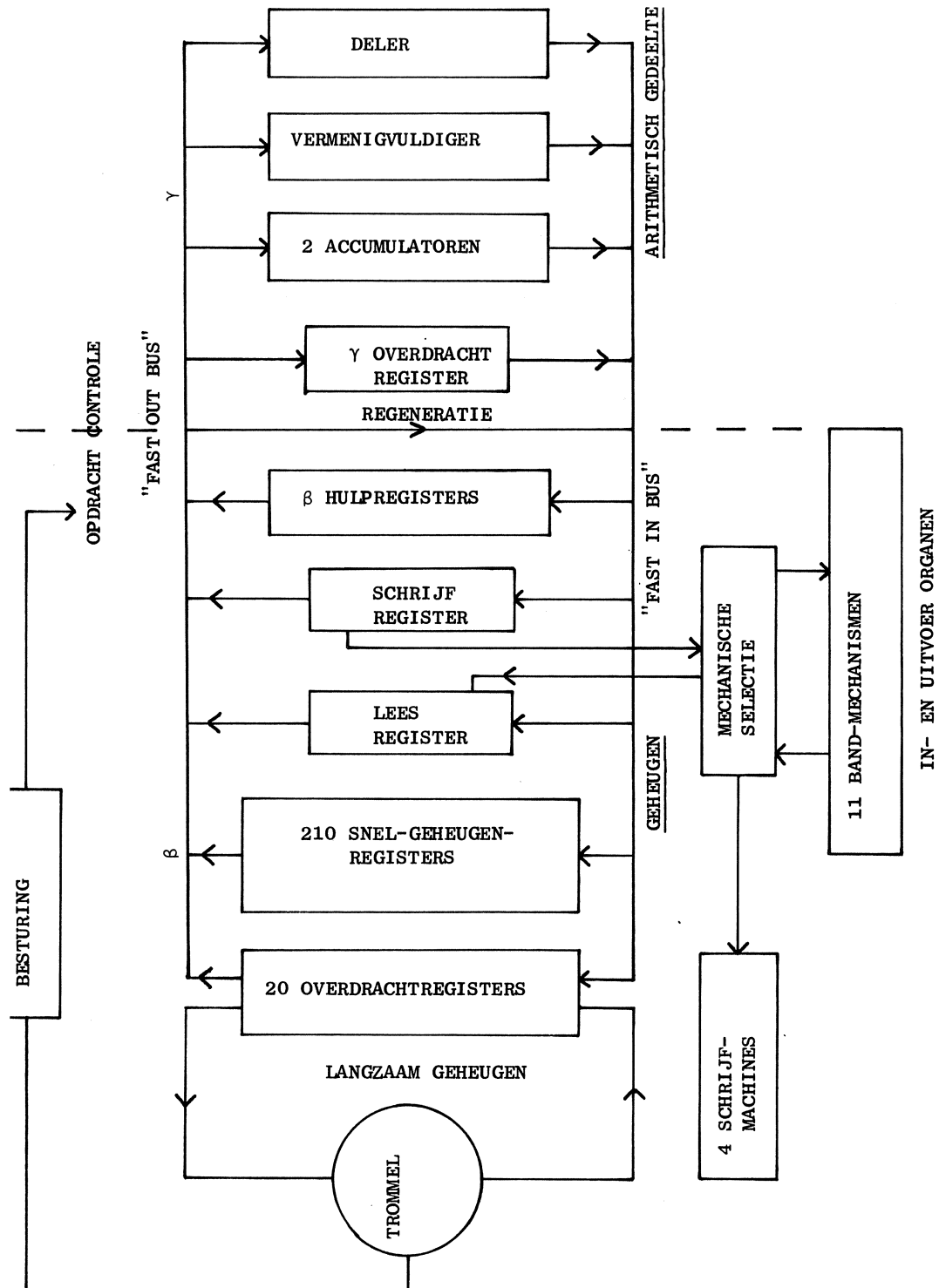
De invoer tot de machine en de resultaten door de machine verkregen, worden op magnetische band geplaatst. Banden kunnen beschreven worden door a) de schrijfapparatuur van de machine, b) de bandschrijftafel, c) de programmeermachine. Getallen worden door alle drie eenheden op dezelfde wijze geschreven. Getallen door de programmeermachine geschreven, worden gebruikt als programma. Banden kunnen worden gelezen door: a) de bandlezers van de machine, b) de vier schrijfmachines.

De bandschrijftafel, de programmeermachine en de schrijfmachines zijn hulpapparatuur, die onafhankelijk van de machine werken. Zij worden gebruikt om vraagstukken voor te bereiden en de resultaten uit te schrijven. De bandschrijfmachine en de programmeermachine worden met de hand bediend. De schrijfmachines worden onafhankelijk van elkaar door relais gecontroleerd. Een gemiddelde van 20 getaltijden is nodig om een getal van een band te lezen of op de band te schrijven. De bandmechanismen worden pneumatisch bediend. 11 bandmechanismen zijn beschikbaar, die elk met de schrijf- of leesapparatuur of de schrijfmachines kunnen worden verbonden.

Opdrachtgedeelte

Het opdrachtgedeelte van de machine bestuurt de machine door middel van een reeks opdrachten. Elke opdracht bestuurt de machine gedurende een getaltijd. Een geheugencapaciteit voor 10.000 opdrachten is beschikbaar op de magnetische trommel. Elke geheugenplaats wordt aangegeven door een getal, adres. Opdrachten worden gewoonlijk in serie gelezen, met oplopend adres, maar het is mogelijk zulk een serie opdrachten te onderbreken en een nieuwe serie te beginnen bij een ander adres.

Een opdracht bestaat uit drie onafhankelijke gedeelten: α opdracht, β opdracht en γ opdracht. De β opdracht identificeert de β eenheid, de γ opdracht de γ eenheid, de α opdracht bepaalt de richting van overdracht en het teken waarmee de overdracht plaats vindt. Het opdrachtgedeelte van de machine heeft mogelijkheden voor handbediening van de machine.



ARITHMETISCHE METHODES, TEN DIENSTE VAN DE PROGRAMMERING

E.W. Dijkstra

Lezing in het colloquium Moderne Rekenmachines van het Mathematisch Centrum op 28 februari 1953 te Amsterdam.

Waar voorbeelden van methodes van programmering in concreto zullen worden uitgewerkt, zullen we ons bedienen van de oude-ARRA-code, die in het kort samengevat hieronder volgt (de inhoud van het adres x wordt met (x) aangeduid, de twee registers A en S hebben beide de capaciteit van een geheugenplaats, het product en deeltal wordt over beide geschreven, met dien verstande, dat de binaire plaatsen in A overeenkomen met het 2^{29} -voud van de corresponderende plaatsen in S, "A staat links van S").

- 0/x vermeerderd (A) met (x)
- 1/x plaatst (x) in S
- 2/x vervangt (x) door (A)
- 3/x plaatst (S) in x en A
- 4/x plaatst $(x) \times (S) = A$ in A en S (gehele getallen)
- 5/x plaatst $(x) \times (S)$ afgerond in A
- 6/x vermenigvuldigt (A) met 2^x
- 7/x verwijst de besturing naar x , als $(A) \geq 0$
wordt genegeerd, als $(A) < 0$
- 8/x vermindert (A) met (x)
- 9/x communicatie-opdracht
- 10/x plaatst (A) in x en vervangt (A) door 0
- 11/x plaatst (S) in x en vervangt (A) door 0
- 12/x deelt het getal in A en S door (x) , plaatst het
quotiënt in S, de rest in A (gehele getallen)

- 13/x deelt (A) afgerond door (x), plaatst quotiënt
in S, maakt A schoon
- 14/x vermenigvuldigt (A) met 2^{-x}
- 15/x verwijst de besturing naar x, als (A) < 0,
wordt genegeerd, als (A) ≥ 0

Bij al het volgende moet niet uit het oog verloren worden, dat de mate van praktische bruikbaarheid van de methodes in hoge mate afhangt van de specifieke eigenschappen van de betrokken machine. Over het algemeen loont het toepassen van de hier aangestipte methodes meer, naar mate de machine sneller en geheugenruimte krapper zijn.

Het gaat hier om processen, waarin naast de eigenlijke berekening, speciale numerieke processen worden uitgevoerd, waarvan de uitkomsten bepalen, welke opdrachten (van de eigenlijke berekening) vervolgens worden uitgevoerd.

Als meest bekende geval: de telroutine, in de cyclus, die een van te voren bepaald aantal malen n herhaald worde.

Indien het niet de bedoeling is, dat een of meer opdrachten van de cyclus bij herhaling modificatie ondergaan, last men een geheel onafhankelijke telroutine in. Wanneer hetzelfde programma bij verschillende waarden van n dienst moet doen, is men welhaast op deze methode aangewezen. (Het alternatief is: de cyclus $n_{\max}$ maal uitschrijven en de besturing elke keer n blokjes voor het einde van de rij er in laten springen; dit kost veel ruimte en is niet eens altijd mogelijk, bv. als $n_{\max}$ niet bekend is.)

Als voorbeeld volgt ($x < 1$, $n \geq 0$): plaats x^n a in 68 als (65) = x, (69) (66) = n, (67) = 1 zo, dat x in 65, n in 66 blijven staan.

c	0/69	64 is werkruimte, waar de telling wordt bijgehouden
c+1	10/68	
c+2	8/66	
c+3	7/c+11	→ (indien n = 0, en er helemaal niet vermenigvuldigd moet worden)

c+10	→	c+4	10/64	
		c+5	1/68	] vermenigvuldigt (68) met x
		c+6	5/65	
		c+7	10/68	
		c+8	0/64	
		c+9	0/67	
		c+10	15/c+4	→
c+3	→	c+11		

Indien de genegeerde besturingsverplaatsing wel tijd kost, verdient het de aandacht, dat we soms door cyclische verwisseling van de opdrachten in de cyclus zonder ruimteverlies tijdwinst kunnen boeken, door een soms onvermijdelijke ongeconditioneerde besturingsverplaatsing uit de cyclus naar de inloop te verwijzen (zie volgende voorbeeld).

Ondergaat het adres van een opdracht bij herhaling een regelmatige verandering, bv. een constante vermeerdering, dan kan de telling beter - dus zuiniger - aan dit adres worden uitgevoerd, wat in het volgende voorbeeld geïllustreerd wordt:

als men bv. de som van de getallen in 100, 101 t.e.m. 149 in 150 wenst te plaatsen, dan geschiedt dit als volgt:

c		10/150	
c+1		0/c+3	
c+2		7/c+6 ==>	(ongeconditioneerd, omdat we het teken van A kennen)
c+3		0/100	constanten
c+4		0/49	
c+12 ==>	c+5	0/c+8	
c+2 →	c+6	10/c+7	
(c+6)	c+7	(0/...)	
	c+8	0/150	
	c+9	10/150	
	c+10	0/c+7	
	c+11	8/c+4	
	c+12	15/c+5	→

Dit kan vanzelfsprekend ook aan een prettig verlopende numerieke grootte worden toegepast.

Gecomplieerdere tellingen kunnen met een verfining analoog worden uitgevoerd, omdat we nog niet alle aan een tel-routine mee te geven informatie benut hebben. Immers, we begonnen met een startwaarde $-s$ en een increment i , en we extraheerden daaruit n als die waarde, waarvoor $n_i - s = 0$; door $i > 1$ te kiezen, n te definiëren als de laagste waarde, waarvoor $n_i - s \geq 0$, kunnen we aan de twee getallen startwaarde en increment na afloop van de telling nog enige informatie ontfutselen, de grootte $r = n_i - s$, waarmee we 0 gepasseerd zijn. Door bv. r , het "overschot" als increment van een volgende telling, het zojuist als increment gebruikte getal ($> r$!) als startwaarde te gebruiken, kunnen we een nieuwe telling lanceren, enz. Het overschot 0 zal als criterium dienen, dat de laatste telling geweest is.

Het onderstaande programma berekent op deze wijze (gehele getallen)

$$(a)^9 (a+1)^4 (a+2)^7 (a+3)^2 (a+4)^1 (a+5)^4 (a+6)^2 (a+8)^1 \bmod p$$

	c+0	1/c+11	plaatst 1 in S
	1	0/c+12	] plaatst 4/a-1
	2	10/c+17	
			op de plaats der vermenigvuldig
			opdracht
	3	8/c+13	] zet startwaarde s_0
	4	10/65	
			negatief
	5	8/c+14	] zet increment s_1
c+28 →	6	10/64	
			negatief
	7	0/c+11	] hoogt het adres
	8	0/c+17	
	9	10/c+17	
			der vermenigvuldig-
			opdracht 1 op
	10	7/c+21 ==>	
	11	1	
	12	4/a-1	
	13	36191	s_0
	14	3695	s_1
	15	p	

c+23 ==>	16	10/54	]	telling
(c+2,9)	17	(4/a+i)	]	vermenig-
	18	12/c+15		vuldiging
	19	10/66		modulo p
	20	1/66		
c+10 →	21	0/65		
	22	8/64	]	telling
	23	15/c+16 →		test
	24	10/66		bergt overschot r
	25	0/64		zet oude increment
	26	10/65		op de plaats van de startwaarde
	27	8/66		zet -r in A
	28	15/c+6 →		test of einde

Als de n^{de} bewerking a_n maal doorlopen wordt, berekent men s_0 en s_1 met de teruglopende recurrente betrekking

$$s_n = (a_n + 1)s_{n+1} - s_{n+2}$$

waar $s_{N+1} = 1$ en $s_{N+2} = 0$ de beginvoorwaarden zijn, als N de index van de laatste bewerking is.

Opm. Nullen mogen in de reeks niet voorkomen.

De programma's in de bovenstaande voorbeelden zijn meer dan een maal te gebruiken, omdat de voor de inloop benodigde informatie onaan-geroerd blijft staan.

Soms echter kan de bepalende functie van de telroutine worden over-genomen door een, de inloops informatie steeds wijzigende berekening, die, als na afloop van het goede aantal herhaalde bewerkingen de bestu-ring de cyclus verlaat, juist weer de inloops informatie achterlaat zoals en waar deze oorspronkelijk werd aangetroffen.

Eerste voorbeeld: een stuk moet steeds twee maal herhaald worden.

De besturing kome in c+1 in de cyclus

aanvankelijk bevat 66 +1;

d+1	→	c	10/66	
	→	c+1		
		.		
		.		
		.		
		d	8/66	
		d+1	15/c	→
		d+2	10/66	plaats weer +1 in 66.

Tweede voorbeeld: bij de integratie volgens Simpson moeten de functie-waarden alternerend met 4 en 2 worden vermenigvuldigd, m.a.w. de opdrachten 6/2 en 6/1 moeten elkaar afwisselen.

In stede steeds de ene opdracht te juister plaatse in het programma, de andere in de werkruimte te hebben staan en ze steeds te verwisselen, kan men beter een numeriek proces gebruiken, zodat

$$f(6/1) = 6/2 \quad \text{en} \quad f(6/2) = 6/1.$$

Men trekt het snelst de in het programma staande opdracht van de som af.

In beide voorbeelden was de eis, dat we twee getallen alternerend genereerden, en wij konden hier ook volstaan met een lineaire betrekking.

Dit systeem is niet uit te breiden, als de eis is, cyclisch de (gehele) getallen $x_0, x_1, x_2, x_0, x_1, x_2$ etc. te genereren. Wij zouden dit met een quadratische betrekking kunnen doen, algemeen, n grootheden $x_0, x_1, \dots, x_{n-1}$ kunnen met een betrekking van de graad $n-1$ worden gegenereerd ($x_{(p+1) \bmod n} = f(x_p)$ mits er geen twee gelijke waarden onder voorkomen; bv. 1, 2, 1 zou niet door een quadratische betrekking gegenereerd kunnen worden). Hier kan de volgende methode overwogen worden:

Een vast getal delen we steeds door een veranderlijke deler: het quotiënt zij het betrokken getal uit de cyclische groep, als functie van de rest is de volgende deler bepaald. Zo kunnen we gelijke quotiënten, die beantwoorden aan verschillende plaatsen in de cyclus met behulp van de bijbehorende resten van elkaar onderscheiden. Moeten we (1,2,1) cyclisch genereren, dan kan dit, door als deler te gebruiken het tweevoud van de juist achtergelaten rest. Door oplossing van homo-

gene lineaire diophantische vergelijkingen vinden we bv.

als eerste deeltal 17, als eerste deler 14. Immers:

$$17:(4=\underline{1}, \text{ rest } 3; 17:(2 \times 3)=\underline{2}, \text{ rest } 5; 17:(2 \times 5)=\underline{1}, \text{ rest } 7$$

en $2 \times 7 = 14$, onze eerste deler. De grootte van de factor, waarmee de rest vermenigvuldigd moet worden, hangt af van de maximum waarde van het quotiënt, van een waarde uit de cyclus, gedeeld door de volgende. Komt in de reeks ongelukkigerwijs een groot quotiënt voor, dan worden hierdoor de aanvangsconstanten snel groter. Aangezien bij tellingen het "overschot" praktisch analoog is aan de rest, is deze methode ook in die vorm toe te passen.

GELIJKRICHTERS

N.C. de Troye

Lezing in het colloquium Moderne Rekenmachines van het Mathematisch Centrum op 25 april 1953 te Amsterdam.

Gelijkrichters vormen een voornaam onderdeel van rekenmachines. Het is de bedoeling van dit colloquium om na te gaan, wat voor een type gelijkrichter het beste te gebruiken is.

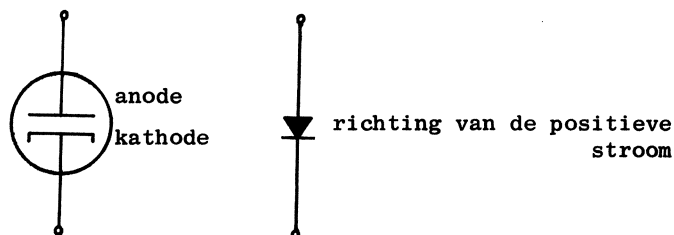
Men kan 2 typen gelijkrichters onderscheiden, namelijk het type dat gebruik maakt van een gloeikathode en het type dat berust op halfgeleiding.

We zullen beide typen nader bekijken, waarbij 3 soorten besproken worden:

- | | |
|-------------------------------------|------------------|
| 1e. gelijkrichter met gloeikathode, | |
| 2e. sperlaaggelijkrichters, | } halfgeleiders. |
| 3e. germaniumgelijkrichters. | |

Gelijkrichters met gloeikathode

Dit type, dat wel in hoge mate bekend is, bestaat uit een gloeikathode + anode. Het geheel is in een glazen ballon ingesmolten en vacuum gezogen. Indien de diode geleidt, is de weerstand $\pm 150\Omega$, indien de diode in de sperstand staat, is de weerstand $\pm 10^7\Omega$. Dit is dus ruwweg een verhouding van $1 : 10^5$.

fig. 1.

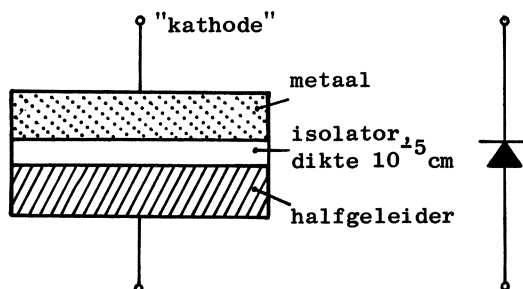
Gelijkrichters, die gebruik maken van halfgeleiders

Sperlaagcellen

Deze gelijkrichters maken gebruik van het verschijnsel, dat bij bepaalde contacten de overgangsweerstand van de ene stroomrichting vele malen groter is dan voor de andere stroomrichting. In 1874 ontdekte Braun dit verschijnsel bij metaalsulfiden, waarop de punt van een metalen draad werd gedrukt. Het effect was slecht reproduceerbaar. Bijna tegelijkertijd vond men, dat 2 koperen plaatjes die in contact met elkaar waren en waarvan één geoxideerd was, hetzelfde verschijnsel vertoonden. Ook dit verschijnsel was slecht reproduceerbaar.

Tegenwoordig heeft men z.g.n. sperlaagventielen ontwikkeld, die goed reproduceerbaar zijn. Deze ventielen zien er als volgt uit:

fig. 2.



De isolator is bijvoorbeeld een oxydehuidje van het metaal dat de kathode vormt. Gebruikt kan worden Al, Zr, Ti of Nb. Het huidje kan ook van hars, zwavel of cellulose zijn.

Het lijkt op het eerste gezicht nogal vreemd, dat deze manier van opbouwen een 2-zijdige functie heeft. Verwacht kan worden, dat de isolator de stroom in beide richtingen zal opheffen. Dat dit niet zo is, kan het volgende duidelijk maken:

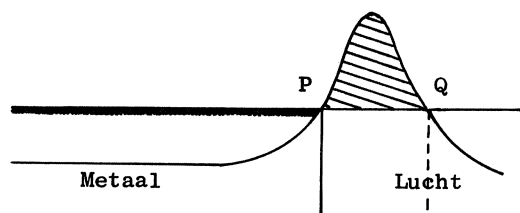
Het vacuum is een perfecte isolator. Weet men op de een of andere wijze electronen in het vacuum te brengen, dan heeft men een geleider. Voegt men aan een isolator op de een of andere wijze electronen toe, dan wordt de isolator eveneens geleidend.

De werking van de sperlaag gelijkrichter is nu als volgt te verklaren. In het metaal bevinden zich vele geleidingselectronen, in de half-

geleider een ± 1000 maal kleinere hoeveelheid. De uittree-arbeid van de electronen is bij het metaal kleiner dan bij de halfgeleider. De electronen zullen dus bij voorkeur van het metaal uit door de isolator heen naar de halfgeleider bewegen. De thermische emissie verklaart echter niet het gehele verschijnsel. Bij toenemende spanning moet dan verzadiging op gaan treden. Dit vindt men echter niet. Bij afkoeling van het ventiel moet de stroom tot nul terug lopen. Men vindt wel een afname, maar de stroom wordt zeker niet nul. Er ligt dus een tweede verschijnsel aan ten grondslag en wel dat van de koude emissie. Legt men boven een metaal een veld aan van 10^7 V/cm dan worden de electronen uit het metaal gerukt. De stromen, die optreden, zijn zeer groot en wel van de orde van 1000 A/cm^2 . Om een veld van 10^7 V/cm op te wekken, gebruikt men een zeer scherp puntje, dat boven het metaal gehouden wordt. Doordat dit puntje een oppervlak van 10^{-8} cm^2 heeft, is de stroom die zal optreden toch nog klein.

De koude emissie is als volgt te verklaren:

fig. 3.



In fig. 3 staat het potentiaalverloop zoals dat zich in een metaal voordoet. Aan de rand treedt een potentiaalsprong op, die we als uittree-arbeid hebben leren kennen. Volgens de klassieke mechanica zal een electron aan de wand gereflecteerd worden. Bij Q is weer dezelfde potentiaal als bij P. Hier kunnen zich weer electronen bevinden, echter niet in het gebied tussen P en Q. Wanneer we de botsing van het electron tegen de potentiaalberg beschrijven in termen van de golfmechanica, dan moeten we zeggen, dat we hier een totale reflectie van materiegolven hebben. Neemt men twee blokken glas, die door een smalle luchtspleet gescheiden zijn en laat men nu in een van de twee blokken een lichtstraal met voldoende grote hoek invallen, dan zal totale reflectie op-

treden. In de lucht treedt ook een electromagnetisch veld op, dat exponentiëel afneemt. In het tweede blok vindt men dus ook straling. Dezelfde redenering kunnen we toepassen op de golf van het electron. Het is dus heel goed mogelijk dat een electron door de potentiaalberg heendringt. Men vindt op grond van deze overwegingen, dat de stroom i gelijk is aan

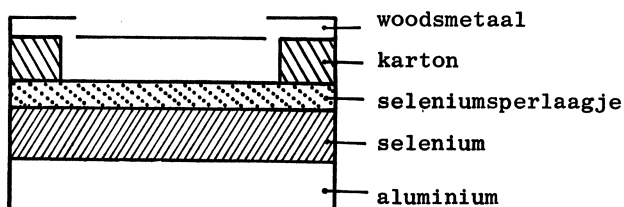
$$i = A \cdot F^2 \cdot e^{-B/F}.$$

B is evenredig met de uittree-arbeid tot de macht $3/2$, F evenredig met V . Dit is moeilijk te beschrijven, want het oxydehuidje is niet overal even dik en puntjes verstoren het veld heel sterk.

De grote overgangsweerstand in de niet doorlatende richting moet verklaard worden door aan te nemen, dat in deze richting veel kleinere koude emissie optreedt. Dat de koude emissie in de halfgeleider veel kleiner is dan in het metaal, kan men verklaren door te zeggen, dat de bewegelijkheid in het metaal groter is dan in de halfgeleider. Per tijdseenheid kunnen dus meer electronen uit het metaal dan uit de halfgeleider treden. Ten tweede is de uittree-arbeid verschillend en ten derde kan men zorgen, dat aan het ene materiaal meer puntjes zitten dan aan het andere, zodat de veldsterkte ook verschillend is voor verschillende richtingen.

De seleniumcellen, die op het Mathematisch Centrum gebruikt worden, hebben de volgende constructie.

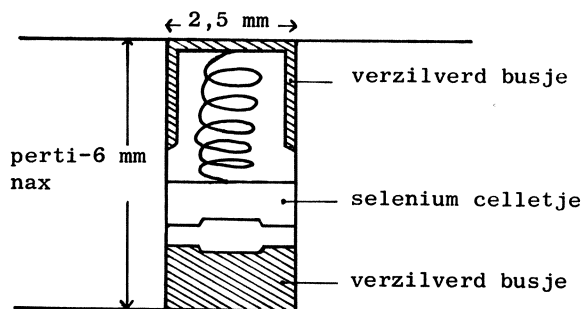
fig. 4.



Doordat de eigenschappen van het selenium veranderen als er druk op gezet wordt, is de constructie zodanig, dat de druk alleen op de plaats van het kartonnen ringetje komt. Het geleidende oppervlak ondervindt dus geen last van druk.

Het celletje wordt als volgt gemonteerd:

fig. 5.



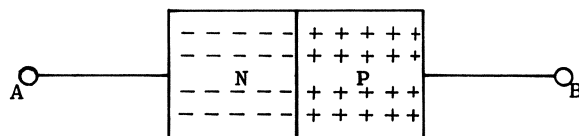
Op het pertinax kan men nu eenvoudig de bedrading schilderen met zilververf. Indien de seleniumcel geleidt, is de weerstand $\pm 6 \text{ K}\Omega$. De sperweerstand $\pm 600 \text{ K}\Omega$. Dat is dus een verhouding van $1 : 10^2$. De seleniumcel is dus aanzienlijk slechter dan de boven besproken diode. Een voordeel is, dat de cel klein is en geen gloeistroom nodig heeft. Een bezwaar is de vrij grote capaciteit ($\pm 70 \text{ pf}$) waardoor de elektrische schakelsnelheid klein wordt ($50 - 80 \text{ kc}$).

Germaniumcellen

Een gelijkrichtcel, die klein is, geen gloeistroom verbruikt en ook snel is, is de OA 55 van Philips. Het principe van deze cel is totaal anders dan dat van de seleniumcel.

We zullen heel in het kort de werking nagaan. Germanium is 4-waardig. Indien men een 3-waardig metaal toevoegt, dan ontstaat er een z.g.n. covalente binding. Het is duidelijk, dat het germanium 1 positieve lading overhoudt. Voegt men een 5-waardig metaal toe, dan ontstaat een covalente binding, waarbij een electron overblijft. In het eerste geval hebben we P-germanium, in het tweede geval N-germanium. Het eerste heeft z.g.n. gatengeleiding, het tweede electronen geleiding. Voor ons is van belang het geval waarbij een plaatje P-germanium in contact gebracht wordt met een plaatje N-germanium.

fig. 6.



Legt men bij A een negatieve spanning en bij B een positieve spanning aan, dan worden bij A electronen in het N-germanium gebracht. Deze lopen door het P-germanium naar B. Bij B worden "gaten" in het P-germanium gebracht en deze lopen door het N-germanium naar A. Er loopt dus een stroom. Draait men de polariteit om dan loopt er geen stroom. De schakeling staat dan in de sperstand. We hebben dus een gelijkrichtende werking gekregen.

De OA 55 heeft een weerstand van $\pm 300 \Omega$ bij geleiding. De sperweerstand is $600 \text{ K}\Omega$; dus een verhouding van 1 : 2000. Deze cel is dus aanmerkelijk beter dan de seleniumcel. De capaciteit is $\sim 1 \text{ pf}$. De schakelsnelheid wordt bepaald door de bewegelijkheid van de gaten. Deze lopen langzaam vergeleken met de electronen. Bij plotseling aanschakelen snellen de electronen weg, de gaten komen langzaam op dreef. Doordat de electronen onmiddellijk gaan lopen, is er dus ook onmiddellijk stroom. Schakelt men plotseling de stroom af, dan verdwijnen de electronen weer meteen, maar de gaten lopen nog langzaam door naar A. De stroom neemt dus langzaam af. Het gevolg is, dat een puls een steile voorflank en een flauwe achterflank heeft.

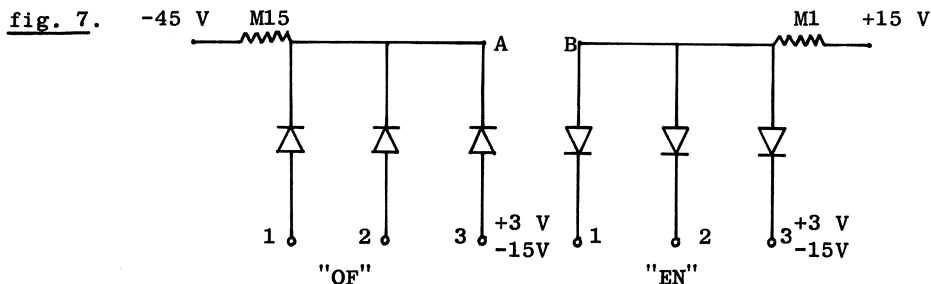
Geven we nog een kort overzicht van alle eigenschappen, dan zien we dat de OA 55 het best te gebruiken is als er kleine stromen geschakeld moeten worden. De cel is klein en heeft geen gloeistroom nodig.

Type	Voorw. weerst.	Tegenw. weerst.	Verh.	Cap.	Snelheid	Temp. afh.	Dim.	Gloeistr.
Diode	150Ω	$10^7 \Omega$	$\pm 10^5$	3 pf	zeer groot	neen	groot	ja
Selenium	6 K Ω	600 K Ω	10^2	70 pf	klein	ja	klein	neen
OA 55	300Ω	600 K Ω	$2 \cdot 10^3$	1 pf	groot	ja	klein	neen

Schakelingen

Voor het gemak gaan we uit van een geïdealiseerde karakteristiek. We zeggen dat de celletjes of volkomen geleiden of volkomen sperren.

Men kan nu de celletjes op verschillende wijzen schakelen. Men verkrijgt dan 2 soorten, n.l. "EN" en "OF" schakelingen.



"OF" schakeling. Indien òf punt 1 òf punt 2 òf punt 3 van -15 V naar +3 V stijgt, dan stijgt eveneens punt A naar 3 Volt.

"EN" schakeling. Indien punt 1 van -15 V naar +3 V stijgt, dan blijft er nog stroom lopen door celletjes 2 en 3. Indien èn punt 1 èn punt 2 èn punt 3 van -15 V naar 3 V stijgt dan stijgt eveneens punt B van -15 V tot +3 V.

Een combinatie van "EN" en "OF" schakeling is mogelijk. Voor een arithmetisch orgaan kunnen we de volgende codering opstellen:

A_n	M_n	C_{n-1}	$\sum_n$	C_n
0	0	0	0	0
1	0	0	1	0
0	1	0	1	0
1	1	0	0	1
0	0	1	1	0
1	0	1	0	1
0	1	1	0	1
1	1	1	1	1

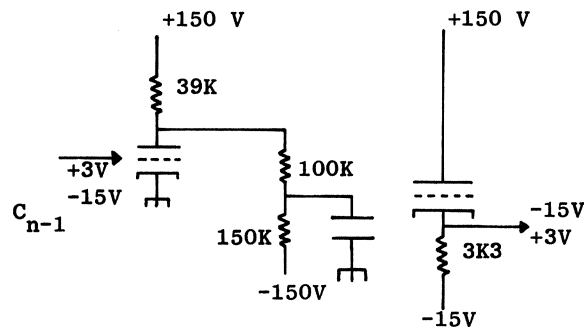
Passen we hierop de relais-algebra toe, dan vinden we:

$$\sum_n = A_n \bar{M}_n \bar{C}_{n-1} + \bar{A}_n M_n \bar{C}_{n-1} + \bar{A}_n \bar{M}_n C_{n-1} + A_n M_n C_{n-1}.$$

$$C_n = A_n M_n \bar{C}_{n-1} + A_n \bar{M}_n C_{n-1} + \bar{A}_n M_n C_{n-1} + A_n \bar{M}_n C_{n-1} \\ = A_n M_n + A_n C_{n-1} + M_n C_{n-1}.$$

Het . teken komt overeen met de "OF" schakeling, het + teken met de "EN" schakeling. We kunnen dus nu gemakkelijk $\sum_n$ en C_n vormen. Voor $\sum_n$ is $\bar{C}_{n-1}$ nodig. Deze maken we uit C_{n-1} door deze door de volgende schakeling te sturen:

fig. 8.



In Eindhoven hebben we bovenstaande $\sum$ en C schakeling gemaakt. Deze liep nog voor één bit betrouwbaar tot 1 MC. Daarboven veranderde het spanningsblokje op een dusdanige wijze, dat het niet meer te gebruiken was.

REKENMETHODES VOOR AUTOMATISCHE REKENMACHINES

A. van Wijngaarden

Lezing in het colloquium Moderne Rekenmachines van het Mathematisch Centrum op 30 mei 1953 te Amsterdam.

Een van de merkwaardige ervaringen, welke men opgedaan heeft bij de reflectie over het gebruik van automatische rekenmachines is, dat de verhouding tussen de benodigde tijd voor het uitvoeren van verschillende bewerkingen, als bijv. optelling, vermenigvuldiging, deling, opzoeken, opbergen, geheel verschillend is van die welke men kent uit het rekenen uit het hoofd, op papier of met tafelrekenmachines. In het algemeen kan men zeggen, dat iedere vooruitgang in de digitale reken-techniek, als het invoeren van het papier, tafelrekenmachine en automatische rekenmachine het uitvoeren van de verschillende bewerkingen in sterk verschillende mate heeft versneld. Bij het uit het hoofd rekenen is de snelheid van opzoeken en wegbergen van een getal zeer groot t.o.v. de rekensnelheid. Men komt daaraan tegemoet door in een betrekkelijk hoog talstelsel (10 of 100) te rekenen en de meest voorkomende berekeningen in de vorm van tafels van optelling en vermenigvuldiging uit het hoofd te leren, d.w.z. permanent op te bergen in het geheugen, zodat ze met grote snelheid door opzoeken ter beschikking staan. Ongeveer hetzelfde geldt voor rekenen op papier. De opzoektijd in tabellen is al aanzienlijk ongunstiger t.o.v. de rekestijd, hoewel het gebruik van productentafels of logarithmentafels nog aanzienlijke versnelling van het rekenproces ten gevolge heeft.

De invoering van de tafelrekenmachine heeft de balans grondig gewijzigd. De rekensnelheid is hiermede zo geweldig toegenomen, dat heel vaak de rekenaar niet meer tracht het aantal rekenbewerkingen minimaal te maken, maar wel het neerschrijven van tussengegevens en het opzoeken in tabellen. De grote verdienste van de ponskaarten-

machines ligt dan ook niet zo zeer in het vergroten van de rekensnelheid als wel in de gezonde combinatie van reken- en administratieve capaciteiten.

Bij de moderne automatische machines zijn de verhoudingen ten slotte nog eens sterk veranderd ten gunste van de rekensnelheid versus de snelheid van administratieve handelingen. Een van de resultaten is dat men veelal functies van een of meerdere variabelen zal berekenen, d.w.z. herleiden tot een aantal elementaire bewerkingen, liever dan ze te interpoleren uit in het geheugen opgeslagen tabellen. Een extreem voorbeeld hiervan is het berekenen van x/y door middel van vermenigvuldigingen, etc., zodat de machine geen ingebouwde deler behoeft te hebben. Het omgekeerde proces, nl. vermenigvuldiging door deling is uit de biologie bekend.

Een en ander heeft aanleiding gegeven tot een hernieuwde belangstelling in de theorie van de repetitieve en iteratieve rekenprocessen. Bij beide is essentieel, dat de berekening bestaat uit een aantal cycli van eenzelfde berekening, waarbij de begingegevens van een cyclus de resultaten van de vorige cyclus zijn. De begingegevens van de eerste cyclus zijn dan naast vaste gegevens eventueel zg. schattingen. We noemen een repetitief proces iteratief als bij iedere cyclus de inverse berekening welke eenvoudig uit te voeren moet zijn, als indicatie gebruikt wordt voor de graad van precisie. Binnen zekere grenzen is een zodanig proces fourterstellend. Een vereiste van het bestaan van een dergelijk proces is het ter beschikking hebben van de definiërende vergelijking in inverse vorm. Als voorbeeld beschouwen wij de deling:

$$x = a^{-1}.$$

Hieruit volgt:

$$f(x) = x^{-1} - a = 0.$$

Deze vergelijking kan iteratief worden opgelost, bijv. met de methode van Newton, dus

$$x_{n+1} = 2x_n - ax_n^2, \quad x = \lim_{n \rightarrow \infty} x_n$$

waarvoor wij gemakkelijk schrijven:

$$x \leftarrow 2x - ax^2.$$

Een eigenschap van een iteratief proces als dit is, dat men direct uit het proces het antwoord kan aangeven. Daarom behoeft men slechts het teken $\leftarrow$ te vervangen door $=$. Men verkrijgt in dit geval dan

$$x = 2x - ax^2,$$

waarvan de oplossingen zijn $x = a^{-1}$, $x = 0$ en $x = -\text{sign } a.00$. Het enige probleem wat overblijft, hoe men de begingegevens moet kiezen, opdat men het gewenste antwoord, hier dus a^{-1} verkrijgt. In dit geval is dit gemakkelijk, in het algemeen uiterst moeilijk.

Bij een repetitief proces in strikte zin (d.w.z. niet iteratief, wat verder zonder meer repetitief genoemd wordt) liggen de zaken geheel anders.

Bijv. kan een repetitief proces voor de deling

$$x = (1 - a)^{-1}, \quad |a| < 1$$

afgeleid worden uit de identiteit:

$$x = (1+a)(1+a^2)(1+a^4)(1+a^8) \dots \quad 1)$$

Het proces ziet er dus als volgt uit:

$$x_{n+1} = x_n(1 + y_n); \quad x_0 = 1, \quad x = \lim_{n \rightarrow \infty} x_n$$

$$y_{n+1} = y_n^2, \quad y_0 = a.$$

Er is nu geen kwestie van, hoe de beginwaarden gekozen moeten worden. Integendeel de keuze hiervan wordt bepaald door de gegevens van het vraagstuk. In het proces zelf komen deze gegevens verder niet voor. Het proces is dus zeker niet fourtherstellend. Het convergentiegebied daarentegen is van te voren aan te geven. Het is helemaal echter niet direct in te zien aan de hand van het proces, wat de uitkomst zal zijn.

1) P.E. Ludgate, On a proposed analytical engine, Sci. Proc. Roy. Dublin Soc., XII, p. 77-91, 1909.

Bij vele repetitieve processen vereist het bepalen van het antwoord als functie van de begingegevens zelfs een grote mate van vindingrijkheid.

Ook het uitrekenen van machtreeksen en kettingbreuken kan worden geïnterpreteerd als het uitvoeren van een repetitief proces. Speciaal over de kettingbreuken willen wij het een en ander onderzoeken, omdat dit terrein voor velen minder bekend is dan dat van de machtreeksen en omdat het voor de toepassing van automatische rekenmachines van groot belang is. ²⁾

Voor de kettingbreuk gebruiken wij de volgende notitie:

$$F_n = \frac{a_1}{b_1} + \frac{a_2}{b_2} + \frac{a_3}{b_3} + \dots + \frac{a_n}{b_n}.$$

Allereerst zou men kunnen menen, dat het toepassen van kettingbreuken praktisch onmogelijk zou zijn op machines zonder deler, met het oog op het grote aantal delingen. Maar men leidt gemakkelijk af, dat

$$F_n = \frac{A_n}{B_n},$$

$$A_{k+1} = b_{k+1} A_k + a_{k+1} A_{k-1}, \quad A_{-1} = 1, \quad A_0 = 0,$$

$$B_{k+1} = b_{k+1} B_k + a_{k+1} B_{k-1}, \quad B_{-1} = 0, \quad B_0 = 1.$$

De partiële tellers A_k en de partiële noemers B_k kunnen dus met hetzelfde repetitieve proces worden bepaald, slechts onderscheiden door de beginwaarden en ter bepaling van F_n is tenslotte slechts een enkele deling noodzakelijk. Wanneer de deling echter niet meer tijd in beslag neemt dan de vermenigvuldiging is door directe deling natuurlijk een enorme besparing mogelijk, omdat een deling plus een optelling in de plaats komt van vier vermenigvuldigingen plus twee optellingen.

2) Een goed leerboek is: H.S. Wall, Analytic theory of continued fractions, van Nostrand, 1948.

Het interessantst zijn natuurlijk de oneindige kettingbreuken.
De meest fundamentele kettingbreuk is die van Gauss voor de hypergeometrische functie, die voor $|z| < 1$ wordt gedefiniëerd door

$$F(a, b, c; z) = 1 + \frac{ab}{1!c} z + \frac{a(a+1)b(b+1)}{2!c(c+1)} z^2 + \dots \quad (c \neq 0, -1, -2, \dots)$$

en verder door analytische voortzetting. Door voor a , b en c bijzondere waarden te kiezen krijgt men allerlei elementaire functies, bijv.:

$$zF(1, 1, 2; -z) = \log(1+z),$$

$$F(-k, 1, 1; -z) = (1+z)^k,$$

$$zF\left(\frac{1}{2}, \frac{1}{2}, \frac{3}{2}; z^2\right) = \arcsin z,$$

$$zF\left(\frac{1}{2}, 1, \frac{3}{2}; -z^2\right) = \arctan z.$$

Vervangt men z door z/a en laat men dan a tot oneindig naderen, dan krijgt men de functie met de machtreeks

$$G(b, c; z) = 1 + \frac{b}{1!c} z + \frac{b(b+1)}{2!c(c+1)} z^2 + \dots,$$

zodat bijv.

$$G(0, c; z) = 1,$$

$$G(c, c; z) = e^z.$$

De kettingbreuk van Gauss is nu:

$$\begin{aligned} 1 - \frac{F(a, b, c; z)}{F(a, b+1, c+1; z)} &= \\ \frac{\frac{a(c-b)}{c(c+1)} z}{1} - \frac{\frac{(b+1)(c+1-a)}{(c+1)(c+2)}}{1} - \frac{\frac{(a+1)(c+1-b)}{(c+2)(c+3)} z}{1} - \frac{\frac{(b+2)(c+2-a)}{(c+3)(c+4)} z}{1} \\ &= \frac{\frac{(a+2)(c+2-b)}{(c+4)(c+5)} z}{1} - \frac{\frac{(b+3)(c+3-a)}{(c+5)(c+6)} z}{1} - \dots, \end{aligned}$$

welke convergeert voor alle (complexe) z met uitzondering van reële $z > 1$ en eventuele zekere geïsoleerde waarden van z (nl. de polen).

Analoog geldt:

$$1 - \frac{G(b, c; z)}{G(b+1, c+1; z)} =$$

$$\frac{c-b}{c(c+1)} \frac{z}{1} + \frac{b+1}{(c+1)(c+2)} \frac{z}{1} - \frac{c+1-b}{(c+2)(c+3)} \frac{z}{1} + \frac{b+2}{(c+3)(c+4)} \frac{z}{1} - \dots$$

welke voor alle z convergeert behalve natuurlijk van de polen.

Met behulp van deze twee machtsmiddelen kan men nu allerlei kettingbreuken maken. Stelt men bijv. in de laatste ontwikkeling $b = 0$ en laat daarna c tot 0 naderen, dan vindt men

$$1 - e^{-z} = \frac{z}{1} + \frac{z}{2} - \frac{z}{3} + \frac{z}{2} - \frac{z}{5} + \frac{z}{2} - \frac{z}{7} + \dots$$

geldig voor alle waarden van z .

Het is deze ontwikkeling die wij eens heden bezien.

Het karakteristieke van een dergelijke kettingbreukontwikkeling is de goede benadering van het overall karakter van een functie. Vergelijken wij haar met de reeksontwikkeling

$$1 - e^{-z} = z - \frac{z^2}{2!} + \frac{z^3}{3!} - \frac{z^4}{4!} + \dots$$

dan zien wij dat de eerste benadering van beide gelijk is, nl. z . De tweede benaderingen zijn

$$z(1 + \frac{z}{2})^{-1} \quad \text{resp.} \quad z(1 - \frac{z}{2}).$$

Nu is het waar, dat de eerste uitdrukking een pool heeft voor $z = -2$ maar voor positieve z streeft ze naar een constante, terwijl de tweede uitdrukking alleen maar in de omgeving van de oorsprong een goede benadering geeft. Maar ook in de buurt van de oorsprong zelf is de kettingbreuk preferabel. Immers:

$$z(1 + \frac{z}{2})^{-1} = z - \frac{z^2}{2} + \frac{z^3}{4} - \dots$$

wat een betere benadering is dan

$$z - \frac{z^2}{2}.$$

Ter illustratie geven wij voor $z = 1$ de waarden die volgen uit de machtreeks en uit de kettingbreuk met gebruik van verschillend aantal termen:

Aantal termen	Macht-reeks	Kettingbreuk
1	1.0000 0000	1.0000 0000
2	0.5000 0000	0.6666 6667
3	0.6666 6667	0.6250 0000
4	0.6250 0000	0.6315 7895
5	0.6333 3333	0.6321 8391
6	0.6319 4444	0.6321 2435
7	0.6321 4286	0.6321 2025
8	0.6321 1806	0.6321 2054

De juiste waarde $1 - e^{-1} = 0.6321\ 2056$. De kettingbreuk convergeert blijkbaar iets beter.

Een aanzienlijke verbetering kunnen wij bereiken als van te voren afgesproken wordt welk interval z zal doorlopen en welke precisie wordt vereist. Als voorbeeld stellen wij $0 \leq z \leq 0.5$ en de vereiste precisie $2^{-28} = 3.6 \times 10^{-9}$.

Als men nu 5 termen neemt en $z = 0.5$ vindt men: $1 - e^{-\frac{1}{2}} = 0.393470790$, i.p.v. 0.393469341 dus een fout van 1449×10^{-9} . Voor kleinere z is de fout natuurlijk kleiner. De gedachtengang is nu als volgt.

Wij gaan de coëfficiënten 1, 2, 3, 2 en 5 wat veranderen en proberen dit zo te doen, dat wij de gewenste precisie wel bereiken. Een erg systematische manier om dit te doen is lastig te vinden, maar met enige goede wil bereikt men wel een redelijk resultaat. Het eenvoudigste is het veranderen van een enkele coëfficiënt, nl. de 5. Voor $z = 0.5$ kan men gemakkelijk (nl. uit de staart van de kettingbreuk) berekenen, wat die 5 effectief moet worden. Men vindt dan 5.2589. Zou men precies deze waarde nemen, dan krijgt men voor lagere z natuurlijk een fout. Maar dit is minder erg dan het lijkt. Immers de invloed van die 5 wordt van lagere z ook veel minder. De uitkomst is voor kleine z ongevoelig

voor die 5. Daarom is het resultaat toch al een enorme verbetering. Men kan nu enkele waarden in de buurt proberen en die zo bepalen dat de uiteindelijke maximale fout minimaal wordt. Zo bereikt men een precisie van ongeveer 10^{-7} . Dit is echter nog niet goed genoeg.

Om verder te komen rekenen wij als volgt: Neem de coëfficiënten 1, 2 en 3 even vast. Dan kan men aan de hand van de voorgeschreven precisie uitrekenen binnen welke waarden (afhankelijk van z) de resterende noemer $2' + z/5'$ mag variëren. Men kan dan proberen of er een lineaire functie bestaat welke aan de eisen voldoet. Dit blijkt niet het geval te zijn en men gaat nu de 3 zo variëren tot het defect minimaal is. Dan varieert men de eerste 2 nog wat, net zo lang tot alles klopt. De Heer Dijkstra heeft de berekeningen uitgevoerd en het voorlopige resultaat is, dat

$$\frac{z}{1} + \frac{z}{2.00004} - \frac{z}{2.99945} + \frac{z}{1.9889986} - \frac{z}{5.717552887} ,$$

de functie $1 - e^{-z}$ in het interval $0 \leq z \leq 0.5$ benadert met een fout van ten hoogste 2^{-28} .

HET GEBRUIK VAN DE B-TUBE BIJ DE FERRANTI COMPUTERS

H.A. Lauweriër

Lezing in het colloquium Moderne Rekenmachines van het Mathematisch Centrum op 27 juni 1953 te Amsterdam.

De door Williams en Kilburn ontworpen en door Ferranti geconstrueerde computer welke in de Manchester University staat bezit een zogenaamde B-tube welke tot bijzondere vereenvoudigingen bij het programmeren leidt. In het kort gezegd komt het hierop neer dat een instructie, welke door de Control uit het geheugen gehaald wordt niet zonder meer door de computer uitgevoerd wordt, maar dat bij die instructie eerst nog wat opgeteld moet worden en wel de inhoud van het in de instructie gespecificeerde B-register. De daarmee verkregen definitieve instructie wordt dan door de machine uitgevoerd. In het geheugen staan dus slechts voorlopige instructies.

Het is duidelijk dat men, door de inhoud van het B-register te wijzigen, met eenzelfde voorlopige instructie verschillende resultaten kan krijgen. Door i.h.b. het adresgedeelte te wijzigen kan men op gemakkelijke wijze met reeksen getallen in opvolgende adressen manipuleren.

De B-tube bestaat bij de Manchester Computer uit acht registers 0 t/m 7 in elk waarvan een instructie past. Een instructie bestaat hierbij uit drie delen:

Functie	B-register	Adres
F	b	S

Het b-getal geeft aan welk B-register bij de voorlopige instructie opgeteld moet worden. In de regel is de nulde B-lijn altijd leeg, zodat, wanneer $b = 0$ is de voorlopige instructie tevens de definitieve is.

Men kan een B-lijn het gemakkelijkste opvatten als een kleine accumulator waarin optellen en aftrekken, overdracht van en naar het geheugen - zonder schoonmaken - en discrimineren mogelijk is.

Hieruit blijkt dat de B-tube bij uitstek geschikt voor telprocessen is, en dat hierdoor de accumulator aanzienlijk ontlast kan worden.

Historisch is de benaming B-tube op zichzelf interessant: A is de accumulator en C de control tube, D is de drum en E het elektronische geheugen.

De heren Kilburn en Williams zagen hierin een gereede aanleiding voor hun specialité de letter B te bestemmen.

Echter schijnt het idee van de B-tube reeds aan Lady Lovelace, de pleitbezorgster van Charles Babbage - dus zowat 100 jaar geleden, bekend te zijn!

Het gebruik van deze B-tube lichten we toe aan enige voorbeelden van programmeren. Eerst vermelden we enige bijzonderheden van de door ons beschouwde computer.

De computer rekt met woorden van 20 bits en getallen van 40 bits. Een instructie heeft precies de lengte van een woord. Het elektronische geheugen bestaat uit acht C.R.T. elementen met elk 64 woorden. Het totale aantal woorden is dus 512 en men denkt zich deze het gemakkelijkst gerangschikt in 16 kolommen van 32. De bijbehorende adressen bestaan uit een kolomnummer en een rijnummer. De structuur van een elektronische instructie is a.v.

.....	00...	0....	
functie	b-getal	kolom	rij

Een getal beslaat twee opvolgende adressen waarvan alleen het eerste gespecificeerd wordt. De Accumulator bestaat uit een Most significant deel van 40 bits (M) en een Least significant deel van eveneens 40 bits (L). De berekeningen vinden in hoofdzaak in M plaats; L wordt gebruikt bij afronden.

De most significant digit van een getal en van A geeft steeds het teken aan (0 = +; 1 = -).

De natuurlijke plaats van de komma is na de tweede bit. Indien getallen steeds positief geïnterpreteerd zouden worden zouden ze dus in het interval $0 \leq X < 4$ passen. Op de gebruikelijke wijze wordt het interval (2,4) gereserveerd voor de negatieve getallen.

Getallen worden dus mod 4 geïnterpreteerd met een absolute waarde ≤ 2 . Hiermede is het standaard interval $-2 \leq X < 2$.

Het kleinste positieve lange getal is $p = 2^{-38}$ (peuter). Het kleinste positieve korte getal is $k = 2^{-18}$ (kleuter).

Het vermenigvuldig register (R) tenslotte bevat 40 bits en de B-lijnen elk 20 bits.

Bovendien beschikt de computer over een reserve geheugen, de magnetische trommel. Het is mogelijk met een speciale instructie blokken informatie (een of twee C.R.T. elementen) met het elektronische geheugen uit te wisselen.

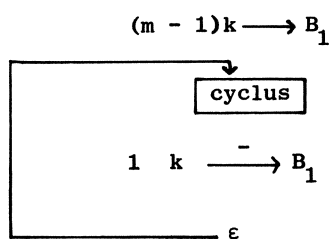
De voornaamste instructies zijn a.v.

P	$s \xrightarrow{+} A$
M	$s \xrightarrow{-} A$
A	$s \longrightarrow A$
α	$A \longrightarrow s$
T	$0 \rightarrow A \rightarrow s$
R	$s \longrightarrow R$
X	$s \times R \xrightarrow{+} A$
Y	$s \times R \xrightarrow{-} A$
C	spring naar s
D	" " " als $A < 0$
E	" " " als $A \geq 0$
π	$s \xrightarrow{+} B$
μ	$s \xrightarrow{-} B$
B	$s \longrightarrow B$
C	$B \longrightarrow s$
ϵ	spring naar s als $B \geq 0$

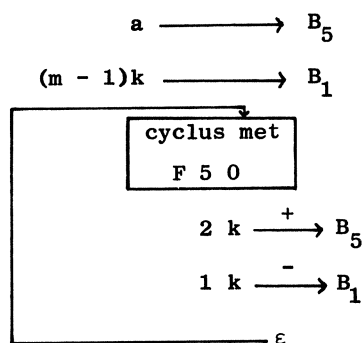
De instructies $\pi\mu B^8$ zijn absoluut, de overige instructies zijn afhankelijk van de door het b-getal gespecificeerde B-lijn.

Zoals reeds is opgemerkt is de B-tube ideaal voor teldoeleinden. De teleenheid is $k = 2^{-18}$. Meestal volgt men de onderstaande schema's of varianten hiervan.

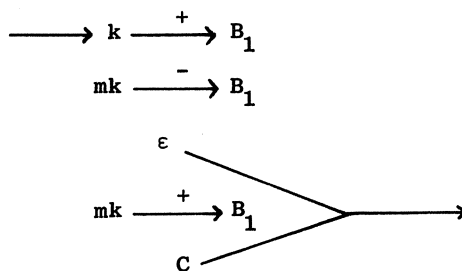
- 1) Repeteer een vaste cyclus m maal

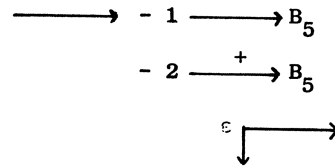


- 2) Repeteer een cyclus met een reeks opvolgende adressen, elk een lang getal bevattend. Het eerste adres is a, het aantal adressen is m.



- 3) Tel modulo m in een B-lijn (in kleuters)



4) Alternator in B_5 

Een aardige toepassing van de B-tube vindt men bij het inroepen van een subroutine, welke op adres 0.0 begint:

	s	B 4 s	
Master	s + 1	C 0 0 0	→
	s + 2		←

Subroutine eindigt met γ 4 0 2

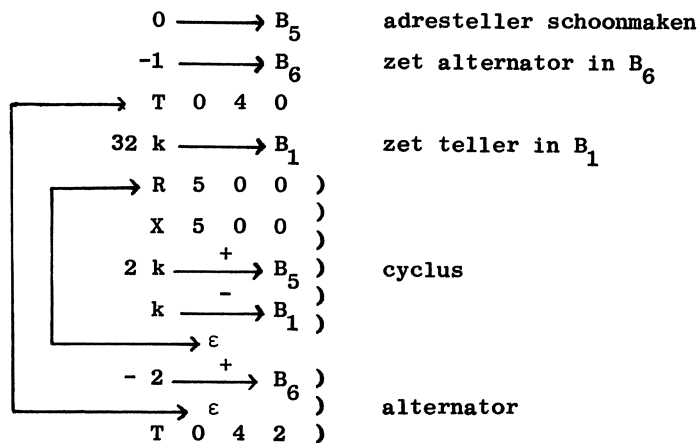
Deze instructie krijgt zijn definitieve vorm a.v.

$$\begin{array}{r}
 B \ 4 \ s \\
 \gamma \ 4 \ 0 \ 2 \\
 \hline
 C - s + 2
 \end{array}
 +$$

Het volgende probleempje kan met behulp van het bovenstaande eenvoudig geprogrammeerd worden.

De 32 getallen a_i bevinden zich in de kolommen 0 en 1, de 32 getallen b_i bevinden zich in de kolommen 2 en 3. Breng $\sum a_i^2$ en $\sum b_i^2$ naar resp. 4.0 en 4.2.

Het volgende schema presteert dit:



Met behulp van de B-tube instructies kan het programmeren aanzienlijk bekort worden. Bij problemen met stelsels lineaire vergelijkingen, b.v. de inversie van een matrix, kan een besparing in geheugenruimte en tijd van b.v. 30% bereikt worden vergeleken met een programma zonder deze faciliteit.

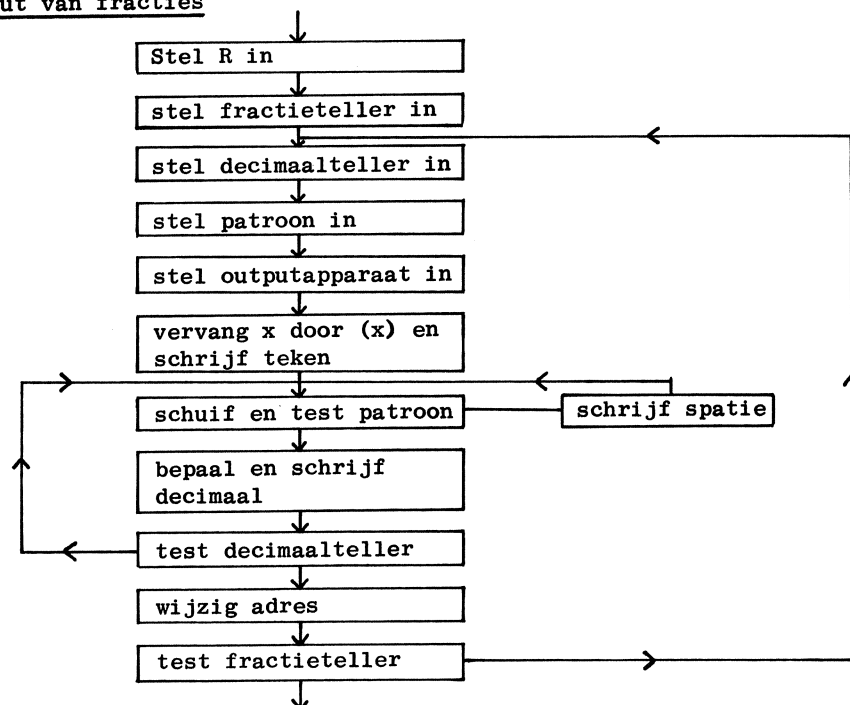
Als voorbeeld van een uitgewerkt programma waarin een aantal B-lijnen gebruikt worden geven we een output subroutine, welke een aantal fracties in opvolgende adressen volgens een willekeurig patroon kolomsgewijs in decimale notatie met teken afdruckt b.v.

```
+ 346 120 567
- 126 586 046
enz.
```

Het hiermede corresponderende patroon is dan

```
0 1 1 1 0 1 1 1 0 1 1 1 -----
```

Schema output van fracties



Output van fracties volgens willekeurig patroon

Aantal fracties in B_1 ; aantal decimalen in B_2 ; eerste adres in B_5 ;
patroon in B_6

0					1				
0	R	0	1	12	Stel R in	0	γ	4	0 2 → terug naar master
1	μ	1	1	3		1	U	0	1 28 ← spatie
2	μ	2	1	4		2	C	0	0 17
3	β	2	1	5		3	0	0	0 1
4	β	6	1	6		4	0	0	0 2
5	B	2	1	5	← begin cyclus	5	-	-	- - aantal decimalen min 1
6	B	6	1	6		6	-	-	- - patroon
7	U	0	1	29		7	-	-	- - werkadressen
8	U	0	1	30		8	-	-	- - "
9	A	5	0	0		9	-	-	- - "
10	E	0	0	15	→ test teken	10	7	31	31 31
11	U	0	1	27	schrijf -	11	31	31	31 31
12	T	5	0	0		12	0	0	0 10
13	M	5	0	0		13	0	0	0 0
14	C	0	0	16		14			
15	U	0	1	26	←	15			
16	T	5	0	0	schrijf +	16			0
17	β	6	1	7	←	17			1
18	π	6	1	7		18			2
19	ϵ	0	1	1	→ test patroon	19			3
20	X	5	0	0		20			4
21	α	0	1	8		21			5
22	B	3	1	8		22			6
23	U	3	1	16		23			7
24	S	0	0	18		24			8
25	Q	0	0	10		25			9
26	T	5	0	0		26			+
27	μ	2	1	3		27			-
28	ϵ	0	0	17	→ test aantal decimalen	28			spatie
29	π	5	1	4		29			wagen terug
30	μ	1	1	3		30			nieuwe regel
31	ϵ	0	0	5	→ test aantal fracties	31			

DE PTERA

W.L. v.d. Poel

Lezing in het colloquium Moderne Rekenmachines van het Mathematisch Centrum op 26 september 1953 te Amsterdam.

De PTERA (PTT Electronische RekenAutomaat) is een automatische rekenmachine. De opzet bij de bouw van deze machine is geweest om de uitvoering zo eenvoudig mogelijk te houden. Dit mocht wel met opoffering aan snelheid maar niet aan mogelijkheden gepaard gaan. De logische opzet vertoont enige bijzonderheden, die hier besproken zullen worden.

Het geheugen bestaat uit een magnetische trommel, waarop zich 64 magnetische aflees- en schrijfkoppen bevinden. Ieder spoor bevat $992 = 31 \times 32$ bits. Er zijn per omtrek namelijk 32 getallen van 31 cijfers. De totale geheugencapaciteit is dus 2048 getallen. Er wordt steeds op slechts een kop tegelijk geschreven of gelezen. Hiervoor is dus slechts een leesversterker en een schrijfpulsinstallatie nodig. De groepering van de bits op de omtrek is als volgt: De cijfers van een getal staan niet als opeenvolgende pulsen op de trommel, maar ze staan zodanig tussen elkaar in, dat eerst alle eerste cijfers van de 32 getallen op een omtrek komen, daarna alle tweede cijfers enz. De omwentelingssnelheid is 2400 toeren per min., d.i. 25 ms per omwenteling. 17 ms worden ingenomen door getalpulsen, de rest wordt de overneemperiode genoemd. De afstand tussen de opeenvolgende pulsen is dus 17 μ s en de afstand tussen de opeenvolgende cijfers van eenzelfde getal is ca. 550 μ s. We hebben dus aan opzoeksnelheid een factor 2 opgeofferd. Elke getalselectie duurt een omwenteling, terwijl bij opeenvolgende plaatsing van de getallen gemiddeld een instructie en een getal per omwenteling geëxtraheerd hadden kunnen worden.

Het rekenkundig orgaan bevat tezamen met de besturing 4 registers welke resp. A (Accumulator), B, C (Control) en R (Relaisregister) heten.

De accumulator bestaat uit 31 eenheden, welke alle slechts kunnen schuiven naar links en optellen. Van deze accumulator wordt tweërlei gebruik gemaakt. Voor optelling wordt hij in serievorm gebruikt. De cijfers uit het geheugen schuiven er een voor een in en de cijfers die al in A stonden, worden rondgekoppeld. Het tekencijfer of wel het meest significante cijfer van het getal komt daarbij voorop! Dat kan omdat A in feite een parallele opteller is.

De vermenigvuldiging vindt parallel plaats, d.w.z. terwijl de vermenigvuldigercijfers in serie uit het geheugen aankomen vindt telkens een plaats opschuiven en een optelling van R al of niet plaats, naarmate het binnenkomende cijfer van de vermenigvuldiger 0 of 1 is. Het tekencijfer van de vermenigvuldiger komt voorop, zodat daar direct bij het begin al rekening mee gehouden kan worden.

Tijdens de vermenigvuldiging ontstaat natuurlijk een dubbele lengte product. Daarom worden dan A en B aan elkaar gekoppeld tot een dubbele lengte register.

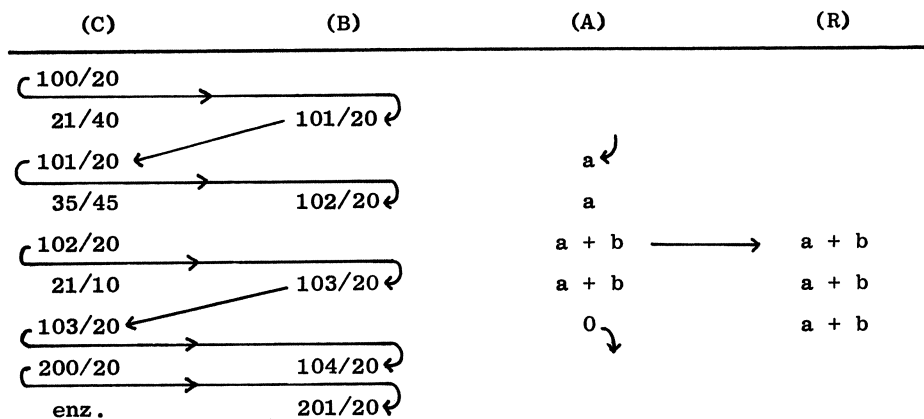
Bij de deling bevindt zich het deeltal in A en de deler in R. Tijdens de bewerking wordt dan telkens getracht R parallel van A af te trekken. Indien het niet gaat, wordt nog gauw de aftrekking ongedaan gemaakt. Het resultaat schuift steeds weer een plaats naar links. Tegelijkertijd schuift het gevormde cijfer van het quotiënt in B en schuift er een cijfer van het uit het geheugen aanvloeiende minst significante gedeelte van het deeltal in A.

De elementaire gang van zaken in een machine met een een-adres-code is altijd dat eerst een instructie geëxtraheerd moet worden en daarna uitgevoerd moet worden. Gedurende deze extractie wordt er eigenlijk niet gerekend, maar wel moet het adres waar straks de volgende instructie gehaald moet worden berekend worden door 1 op te tellen bij het vorig adres. De gedachtengang is nu geweest om voor deze adresseteller het toch in deze phase niet gebruikte B-register te gebruiken. Deze bezit volledige optelmogelijkheid ten behoeve van de vermenigvuldiging. Het C-register bevat steeds de eerstvolgende uit te voeren in-

structie, maar hierbij wordt in deze machine ook het halen van een instructie zelf verstaan! Dat is dus ook een instructie met een adresgedeelte en een opdrachtgedeelte. De gang van zaken bij een eenvoudig program om a en b op te tellen is dan als volgt: (met (x) wordt de inhoud van register x bedoeld)

reg.	adres/opdr.	(21) = a	(35) = b
100	21/40	40 is optellen met schoonmaken	
101	35/45	45 is optellen zonder schoonmaken met overnemen	
102	21/10	10 is opbergen	
103	200/20	20 is haal instructie of wel: spring	

De gang van zaken is hierbij als volgt:



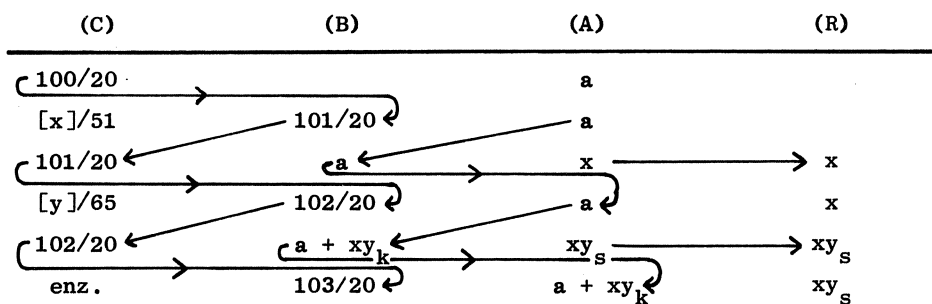
Zoals men ziet is de intern optredende 20-instructie in de adresteller dus ook mogelijk als uitwendige instructie en fungeert dan als sprong. Daarna neemt deze instructie de adrestelling weer intern over. De normale afwisseling: 20-instr. - niet-20-instr. wordt hier dus doorbroken.

Bij de vermenigvuldiging wordt de kop van het product in B en de staart in A gevormd. Opbergen is alleen vanuit A mogelijk. Dus moet er een middel zijn om (B) in A te krijgen. Hiervoor dienen de 50-opdrachten. Deze opdrachten zijn eigenlijk gewone optelopdrachten, maar ze zorgen

voor het inschakelen van een speciale verbinding tussen B en A. Deze opdracht wordt gebruikt als voorbereiding voor de vermenigvuldiging, voor de deling en voor een sprong naar een subprogram. Bijv.

100	[x]/51	Stel dat zich nog een getal a in de accu-
101	[y]/65	mulator bevindt. [x] betekent: het adres
102	enz.	waar zich het getal x bevindt. 51 is de
		code voor voorbereiden, optellen met
		schoonmaken en overnemen.

65 is de code voor vermenigvuldigen zonder schoonmaken met overnemen.
Er gebeurt:



De kop van het product komt tenslotte in A terecht en ook kan de staart nog behouden worden door hem over te nemen in R. Een vermenigvuldigopdracht na een gewone optelopdracht 41 (optellen met schoonmaken en overnemen) gegeven vormt wel de kop van het product maar deze komt niet van B naar A, zodat dan alleen de staart behouden blijft in A.

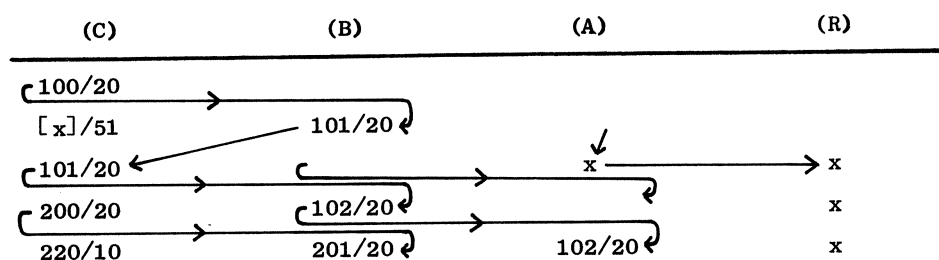
Het vormen van sommen van de meest significante gedeelten van producten is dus hier direct mogelijk. Ook vormen zoals $(ab + c)d + e$ enz. zijn heel gemakkelijk te maken.

Bij de deling vervult de 50-opdracht een soortgelijke rol als bij de vermenigvuldiging. Het quotiënt komt op die manier van B naar A. De rest kan overgenomen worden in R en blijft zo ook behouden.

De toepassing van de 50-opdracht voor een spronginstructie is wel de aardigste mogelijkheid. Om met een getal naar een subprogram te gaan en weer terug te keren naar het punt waar we vandaan gekomen zijn, kunnen we als volgt te werk gaan:

Hoofdprogram		Subprogram	
100	[x]/51	200	220/10 plaats terugkeer- instr.
101	200/20	...	 op 220
102	enz.	(220	)

Er gebeurt:



Nu staat in A dus juist 102/20: spring naar 102 terug en dat wordt opgeborgen op de laatste plaats van het subprogram, zodat dit automatisch terugkeert naar de goede plaats. Bovendien wordt een getal meegenomen in R.

De opdrachten in de PTERA kunnen in 12 hoofdgroepen worden ingedeeld.

Deze zijn:

00: Stop en optellen	01: Ponsband lezen
10: Opbergen	11: Typen en deconverteren
20: Sprong	21: Afronden
30: Test en optellen	31: Conjunctie
40: Optellen	
50: Voorbereiden en optellen	
60: Vermenigvuldigen	
70: Delen	

Elke opdracht heeft een aantal varianten, welke door het tweede cijfer worden aangegeven. Bijv. in kortschrift:

n/40:	$(n) \rightarrow A$	n/44:	$(A) + (n) \rightarrow A$
n/41:	$(n) \rightarrow A \quad (A) \rightarrow R$	n/45:	$(A) + (n) \rightarrow A \quad (A) \rightarrow R$
n/42:	$-(n) \rightarrow A$	n/46:	$(A) - (n) \rightarrow A$
n/43:	$-(n) \rightarrow A \quad (A) \rightarrow R$	n/47:	$(A) - (n) \rightarrow A \quad (A) \rightarrow R$

Dezelfde varianten gelden in principe ook voor de andere hoofdsor-
ten met dien verstande, dat 40, 50, 60 en 70 alle 8 varianten hebben. De
opdrachten onder de 40 vallen uiteen in twee groepen. De even opdrachten
00, 10, 20 en 30, die alleen 4 even varianten hebben en de oneven op-
drachten 01, 11, 21 en 31, die alleen 4 oneven varianten hebben. Alle
oneven opdrachten nemen over in R.

De opdrachten 10, 20, 40, 50, 60 en 70 zijn al ter sprake gekomen.

De testopdrachten zijn tegelijk ook optelopdrachten evenals 40; bo-
vendien slaan de 30 en 34 opdrachten de volgende instructie over als
(R) negatief is en slaan de 32 en 36 opdrachten de volgende instructie
over als (R) positief is.

De opdracht stop is tegelijk een optelopdracht, zodat we voor be-
proeving van een program een 40 door 00 kunnen vervangen voor visuele
inspectie van de op dat moment aanwezige inhoud van A.

21 is afronden. De details zijn oninteressant. Vermelding verdient
wel, dat het bedrag waarmee afgerond wordt ook uit het geheugen moet ko-
men. Bij elke opdracht in deze machine wordt het geheugen gebruikt.

De logische vermenigvuldiging wordt uitgevoerd door de 31-opdrach-
ten. Deze opdracht is nuttig om een stuk uit een getal te knippen.

De invoer van gegevens geschiedt vanaf een geponste band, waarop
zich op elke regel een symbool van 5 bits bevindt. Hiervan worden 4
bits gebruikt om een tientallig (of eigenlijk een 16-tallig) cijfer
voor te stellen. Het vijfde gat heeft een bijzondere functie. Een sym-
bool, dat van een 5e gat is voorzien op de band, wordt geschreven met
een streep erachter. Om de conversie eenvoudig te verzorgen is de lees-
opdracht als volgt ingericht:

n/05: lees een symbool van 4 bits van de band, vermenigvuldig dat
met (n) en tel dat op bij A. Sla de volgende instructie
over alshet cijfer van het vijfde gat is voorzien.

n/01 is hetzelfde, maar met schoonmaken.

Deze opdracht kan op twee manieren gebruikt worden. Om bijv. een getal van drie cijfers te lezen en tegelijk te converteren kan men het volgende program gebruiken.

100	[100]/01	lees het eerste symbool $\times 100$ in
101	[10]/05	tel daarbij het volgende symbool $\times 10$ op
102	[1]/05	tel daarbij het volgende symbool $\times 1$ op

Het aantal cijfers ligt nu vast. Flexibeler is daarom:

100	[10]/60	Vermenigvuldig partiëel resultaat met 10
101	[1]/05	Lees een symbool in

102	100/20	Indien niet gestreept keer terug naar 100
103	enz.	Indien het laatste cijfer gestreept is,

beëindig de cyclus en ga door. Het zal nu meteen duidelijk zijn, waarom de instructies geschreven zijn met het adres voorop. De streep beëindigt het inlezen en met twee volgende leesopdrachten worden de beide (8-talige) opdrachtcijfers erbij ingelezen.

Voor de deconversie staat een even gemakkelijke opdracht ter beschikking.

n/11 betekent: Vermenigvuldig (R) met (n), typ de kop en neem de staart weer over voor de volgende bewerking.

Een representatief deconversie- en typprogram ziet er dus als volgt uit:

100	[x]/41	Neem x op in R
101	10/11	Typ eerste cijfer
102	10/11	Typ volgende cijfer
	enz.	

TELLERS

B.J. Loopstra

Lezing in het colloquium Moderne Rekenmachines van het Mathematisch Centrum op 31 oktober 1953 te Amsterdam.

Onder een teller zullen we verstaan een instrument, dat bepaalde elektrische verschijnselen kan tellen met een van te voren vastgestelde modulus. Over de aard dezer verschijnselen zullen we in bepaalde gevallen zekere veronderstellingen maken en we zullen ons daarbij niet beperken tot het tellen van eenvoudige elektrische impulsen op een invoerlijn.

Wanneer electronici over tellers spreken, denkt men gewoonlijk in de eerste plaats aan het type dat bestaat uit een aantal Eccles-Jordan circuits, uitgerust met condensatoren, dat de eigenschap vertoont op een op verstandige wijze toegevoerde impuls van voldoende steilheid en amplitude van de ene stabiele stand in de andere over te gaan. Wanneer men dan informeert hoe dit eigenlijk komt, krijgt men gewoonlijk een min of meer duidelijk vertoog aan te horen dat men niet begrijpt, en wanneer dergelijke schakelingen weigeren kan het dikwijls uitermate lastig zijn te vinden waar de schoen wringt. De moeilijkheid is, dat een zuiver logische beschrijving van een dergelijk telmechanisme niet wel mogelijk is. Verder is vooral de eis van voldoende steilheid van de drijvende impulsen zeer hinderlijk.

De code waarin deze conventionele tellers tellen, is in het algemeen de gewone binaire code. Door middel van kunstgrepen kan hier wel enige verandering in gebracht worden, doch eenvoudig is dit niet. We kunnen ons afvragen of er codes zijn, welke voor teldoeleinden prettiger eigenschappen bezitten dan de binaire code. M.a.w. we kunnen de vraag stellen welke eisen we aan de code zouden stellen, indien het ons vrij stond deze te kiezen.

Nu is het zo, dat een teller een ding is, dat bij de stappen, die we hem laten doorlopen gebruik moet maken van de informatie die hij zelf bevat, zulks bijvoorbeeld in tegenstelling tot coderende schakelingen, waarbij de uitgang door de ingang i.h.a. volledig bepaald is. Het vervelende van de meeste codes is nu dat men de overgang van de ene code op de naastvolgende logisch gezien in het geheel niet maken kan: immers deze overgang dient te geschieden op grond van de informatie, welke aanvankelijk beschikbaar was, doch die tijdens het proces verdwijnt, zodat men eigenlijk nooit precies weet wat men doet.

We zullen dus trachten de code zo te kiezen, dat geen informatie verloren gaat wanneer we een stap maken. Het is echter duidelijk dat toch tenminste één digit van de teller zal moeten veranderen en iets verliezen we dus wel, maar tegen dit bezwaar bestaat een remedie.

We denken ons daartoe eens een teller die telt in een code waarbij bij iedere overgang slechts één digit verandert en we veronderstellen dat we deze teller drijven met impulsen, welke afwisselend op 2 gescheiden lijnen verschijnen en die we de α - en β -impulsen zullen noemen. We nemen een concreet voorbeeld van een 4-puntscode, welke aan de

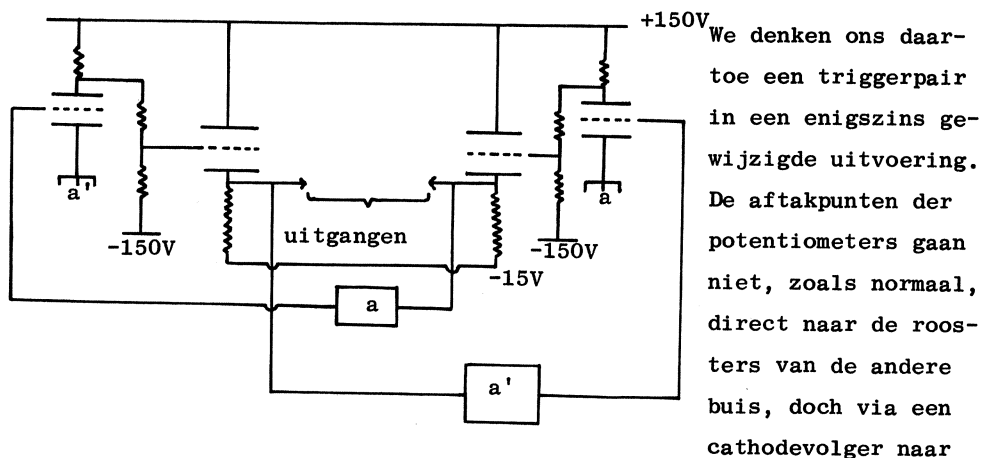
	a	b	c	d	gestelde eisen voldoet; over de moge-
0	0	0	0	0	lijkheid dergelijke codes te maken zul-
1	0	0	0	1	len we straks nog spreken. We spreken
2	0	0	1	1	nu dus af, dat we de overgang van de
3	0	0	1	0	even (nul inbegrepen) naar de oneven
4	0	1	1	0	standen zullen laten plaats vinden op
5	0	1	1	1	α -impulsen, die van oneven naar even
6	0	1	0	1	op β -impulsen.
7	0	1	0	0	Dan is het duidelijk dat de voorwaar-
8	1	1	0	0	den voor een willekeurige overgang,
9	1	1	0	1	bijv. $9 \rightarrow 10$ geheel vastliggen: deze
10	1	1	1	1	overgang vinde plaats indien $a=b=d=1$
11	1	1	1	0	wanneer er een β -impuls verschijnt.
12	1	0	1	0	Dat is dus een der voorwaarden waar-
13	1	0	1	1	onder c van $0 \rightarrow 1$. Aan de conditie
14	1	0	0	1	verandert gedurende de overgang blijk-
15	1	0	0	0	baar niets.

We kunnen dus voor de 4 trigger-pairs a, b, c en d nu precies de voorwaarden opschrijven waaronder ze van $0 \rightarrow 1$ en weer van $1 \rightarrow 0$ moeten gaan. Voor de a is dit bijvoorbeeld al heel eenvoudig:

	$0 \rightarrow 1$	$1 \rightarrow 0$
Voorw:	b', c, d, β'	b, c, d, β'

De notatie zal duidelijk zijn. De noodzaak voor de α - en β -impulsen wordt duidelijk indien men zich realiseert dat we twee eisen stellen: de drijvende impulsen mogen een willekeurige lengte hebben en de α - en β -impulsen mogen elkaar nooit overlappen. Hadden we immers slechts één type impulsen gebruikt, dan zou de teller na de overgang $9 \rightarrow 10$ zodra aan de voorwaarden voor de overgang $10 \rightarrow 11$ voldaan was ook deze overgang hebben gemaakt enz., zolang de drijvende impuls aanwezig is.

We kunnen thans iets zeggen over de wijze waarop we deze teller technisch zouden uitvoeren.



de circuits a en a'. Deze circuits functioneren als en-of schakelingen en hierin zijn de verschillende overgangsvoorwaarden geïncorporeerd.

In woorden gezegd is dus de functie van circuit a' bijv. om een lage uitgang te geven indien of de ingang a' laag is (het zogenaamde houdniveau) of aan één van de voorwaarden waaronder a van $0 \rightarrow 1$ moet veranderen voldaan is. Andere ingangen van het circuit a' zullen dus zijn bv. b, c, d, α , β . Analooq voor a. De opbouw van dergelijke schakelingen kan zeer geschikt met selenium- of germaniumdiodes gebeuren.

Men kan gemakkelijk bewijzen dat de code welke in ons voorbeeld werd gebruikt en waarvan de opbouw duidelijk zal zijn, is uit te breiden tot een willekeurig aantal digits. Er geldt:

$$d_{pq} = \left[\frac{p+2^q}{2^{q+1}} \right] - 2 \left[\frac{p+2^q}{2^{q+2}} \right],$$

waarin p het nummer van de rij en q het nummer van de kolom is.

We zien immers dat de wisselingen in de k^{de} kolom (van rechts naar links) plaats vinden bij overgang van de $p^e = \{2^{k-1}(1+2n)\}^e$ rij naar de volgende.

Willen wij dus weten welke digit in een bepaalde rij p wisselt, dan lossen we op: $2^{k-1} = \frac{p}{1+2n}$. Als p oneven is moet $k = 1$ zijn en daaruit volgt dan n . Als p even is delen we alle gemeenschappelijke factoren 2 aan beide kanten uit en bepalen k en n verder op dezelfde wijze. We vinden dus in iedere rij een en slechts een wisselende digit.

Verder kan men inzien dat codes van dit type voor elke even telcyclus had geconstrueerd kunnen worden, uitgaande van het zojuist besproken soort. Om aan te geven hoe dit gebeurt, behandelen we als voorbeeld een cyclus van tien stappen. We kunnen daarvoor bijvoorbeeld nemen de codes 0 t/m 7, 8 en 15. De vraag hoeveel codes van dit type er bij elke gestelde opgave zijn is echter voor zover mij bekend, niet te beantwoorden. Wel is duidelijk dat cyclische wisseling der rijen of inversie van een of meer kolommen de genoemde eigenschappen niet aantasten, maar hiermede zijn de mogelijkheden niet uitgeput.

Dikwijls beperkt het probleem van het tellen van impulsen zich niet tot dit tellen en het verstrekken van een bijv. visuele indicatie alleen, doch willen we met de uitgangen van onze teller nog iets anders doen. Als voorbeeld noemen we het geval waarin we voor elke stand van de teller een afzonderlijke lijn willen activeren. In dat geval loopt men de kans, door ongelukkige keuze der tellercode, zeer veel apparatuur te moeten gebruiken voor de uitcodering van deze signalen. Het is dan dikwijls beter een telcode te gebruiken waaruit de gewenste signalen met geringe moeite verkregen kunnen worden, zelfs indien daardoor de teller meer trappen zou moeten hebben en gecompliceerder van constructie zou worden.

In het bovengenoemde voorbeeld zou een erg plezierige code bijv. zijn:

0	0	0	0	0	Een teller dus van 4 trappen met 8 stan-
1	1	0	0	0	den, waarbij echter iedere stand aan
2	1	1	0	0	slechts 2 digits is te herkennen, in
3	1	1	1	0	plaats van aan 3, zoals het geval zou
4	1	1	1	1	zijn geweest als we voor de telling
5	0	1	1	1	een driepuntscode hadden gebruikt.
6	0	0	1	1	Welke oplossing in een bepaald geval de
7	0	0	0	1	minimale hoeveelheid apparatuur zal kos-
					ten, hangt natuurlijk o.a. af van de wij-
					ze waarop de uitcodering geschiedt.

We willen er tenslotte op wijzen, dat het in het voorgaande besproken type teller met 2 ingangslijnen in de grond van de zaak eigenlijk niets anders is dan wat we gewoonlijk een ring noemen: een configuratie van geheugenelementen, die volgens bepaalde condities door zijn verschillende mogelijke standen wordt gestuurd. Het is hierbij helemaal niet nodig zich tot twee ingangssignalen en de uitgangssignalen van de elementen zelf te beperken. Als teller gaan deze ringen pas werken, wanneer we ons tot een vast stel ingangssignalen bepalen, waartussen bovendien nog een zeker verband bestaat, zoals in ons geval gegeven wordt door de eis dat de drijvende impulsen afwisselend op de ene en op de andere lijn zullen voorkomen. Een ring kan echter soms met voordeel nog op andere wijze geconstrueerd worden door gebruik te maken van zg. multiplets.

ENIGE HULPMIDDELEN BIJ HET OPLOSSEN VAN BINAIRE VERGELIJKINGEN EN TOEPASSINGEN DAARVAN

C.S. Scholten

Lezing in het colloquium Moderne Rekenmachines van het Mathematisch Centrum op 28 november 1953 te Amsterdam.

Wanneer we ons bepaalde problemen stellen met betrekking tot binaire grootheden, tot welke oplossing wij gebruik denken te maken van wat wij zouden kunnen noemen "binaire algebra", dan resulteren onze onderzoekingen in een aantal binaire vergelijkingen, die wij graag zouden oplossen. Met name willen wij veelal graag weten of er in deze vergelijkingen een stel grootheden $x_1, \dots, x_n$ is aan te wijzen, dat wij zinvol als onafhankelijk veranderlijken kunnen beschouwen, d.w.z. dat zij inderdaad 2^n verschillende waarden combinaties kunnen aannemen en indien dit het geval is op welke wijze de overige grootheden van $x_1, \dots, x_n$ afhangen. Wij willen nu aangeven op welke wijze wij dit enigszins systematisch kunnen onderzoeken.

We denken ons het stelsel vergelijkingen gegeven in de gedaante

$$P_h(x_1, \dots, x_n) = 0 \quad (h = 1, \dots, m),$$

waarbij P_h een polynoom zal zijn van de eerste graad in $x_1, \dots, x_n$ elk afzonderlijk, met in eerste instantie willekeurige (d.w.z. niet noodzakelijk binaire) coëfficiënten (polynomen van hogere graad worden uiteraard direct tot de hier besproken teruggebracht wegens $x^n = x$). Vervolgens vermenigvuldigen wij elke in deze polynomen voorkomende term met de volledige "switching function" van de in die term ontbrekende variabelen. Deze gaat daarbij over in een 2^1 -term, wanneer 1 het aantal ontbrekende variabelen is. (Onder volledige switching function in b.v. $y_1 y_2$ verstaan wij $y_1 y_2 + y_1 y_2' + y_1' y_2 + y_1' y_2' = 1$. Elke volledige switching

function is in getalwaarde gelijk aan 1).

Door nu in elk polynoom gelijksoortige termen bij elkaar te nemen gaat ons stelsel over in:

$$Q_h(x_1, \dots, x_n) = \sum_{i=0}^{2^n} a_{hi} (x_1, \dots, x_n)_i = 0 \quad (h=1, \dots, m),$$

waarbij met $(x_1, \dots, x_n)$ die term bedoeld wordt, die gelijk aan 1 is, wanneer $x_1, \dots, x_n$ juist het binair geschreven getal i voorstellen. De coëfficiënten a_{hi} zijn nog steeds niet binair.

Door te bedenken dat een dergelijk polynoom alleen dan gelijk is aan nul wanneer alle termen nul zijn, leidt ons stelsel tot

$$(x_1, \dots, x_n)_{p_q} = 0 \quad (q=1, \dots, s)$$

of

$$\sum_{r=s+1}^{2^n} (x_1, \dots, x_n)_{p_r} = 1.$$

Hierbij vormen de getallen $p_1, \dots, p_{2^n}$ een permutatie van de waarden $0, \dots, 2^n-1$.

We zien dat we ons stelsel vergelijkingen hebben teruggebracht tot een vergelijking in de standaardvorm

$$\sum_{j=0}^{2^n-1} b_j (x_1, \dots, x_n)_j = 1 \quad (1)$$

waarbij de b_j 's hier wel binaire grootheden zijn.

Uit deze vergelijking kunnen we al direct een belangrijke conclusie trekken. We schrijven hem n.l. met enigszins gewijzigde notatie als

$$\sum_{j=0}^{2^{m+n}-1} b_j (x_1, \dots, x_n, y_1, \dots, y_m)_j = 1 \quad (2)$$

en bewijzen de volgende

Stelling: De nodige en voldoende voorwaarde voor oplosbaarheid van deze vergelijking in de vorm

$$y_i = \sum_{j=0}^{2^n} a_{ij}(x_1, \dots, x_n)_j \quad (i = 0, \dots, 2^m)$$

is dat elke uitdrukking $(x_1, \dots, x_n)_k$ juist eenmaal onder het $\sum$ -teken in het linkerlid van (2) voorkomt. Dit houdt dus o.m. in dat het linkerlid een 2^n -term is.

Ontbreken b.v. alle termen met $x_1, \dots, x_n$, dan wordt voor $x_1 = x_2 = \dots = x_n = 1$ het linkerlid van (2) gelijk aan nul, onafhankelijk van de waarden van $y_1, \dots, y_m$. D.w.z. er bestaat blijkbaar een verband tussen $x_1, \dots, x_n$. Deze kunnen dus niet als onafhankelijk veranderlijken dienst doen. Komt $x_1, \dots, x_n$ eenmaal voor, b.v. in de vorm $x_1, \dots, x_n (y_1, \dots, y_m)_i$, dan volgt uit $x_1 = \dots = x_n = 1$: $(y_1, \dots, y_m)_i = 1$ waardoor alle y 's bepaald zijn. Komt $x_1, \dots, x_n$ meermalen voor, dan zijn voor $x_1 = \dots = x_n = 1$ de y 's niet meer volledig bepaald.

Laat b.v. gegeven zijn het stelsel

$$y_1(x'_1x_2 + x_1x'_2) = 0$$

$$y'_1(x_1x_2 + x'_1x'_2) = 0$$

$$y_2(x'_1x'_2) = 0$$

$$y'_2(x'_1x_2 + x_1x'_2 + x_1x_2) = 0.$$

De eerste twee vergelijkingen vermenigvuldigen we met $y_2 + y'_2$, de laatste twee met $y_1 + y'_1$. Er komt dan, wanneer we termen die in x_1 en x_2 niet verschillen bij elkaar nemen:

$$x_1x_2(y_1y_2)' = 0$$

$$x_1x'_2(y'_1y_2)' = 0$$

$$x'_1x_2(y_1y_2)' = 0$$

$$x'_1x'_2(y_1y'_2)' = 0$$

$$\text{of} \quad x_1x_2y_1y_2 + x_1x'_2y'_1y_2 + x'_1x_2y_1y_2 + x'_1x'_2y'_1y'_2 = 1 \quad (3)$$

waaruit volgt: voor $x_1 = x_2 = 1$ is $y_1 = y_2 = 1$ enz. dus:

$$y_1 = x_1 x_2 + x_1' x_2'$$

$$y_2 = x_1 x_2 + x_1 x_2' + x_1' x_2$$

een niet volledige transformatie.

Staat in het linkerlid van (3) ook nog de term $x_1' x_2' y_1 y_2$, dan volgt uit $x_1 = x_2 = 0$ slechts $y_1' y_2 + y_1 y_2' = 1$ d.w.z. $y_1 \neq y_2$. De oplossing luidt dan:

$$y_1 = x_1 x_2 + \lambda x_1' x_2'$$

$$y_2 = x_1 x_2 + x_1 x_2' + x_1' x_2 + \lambda x_1' x_2'$$

waarbij dan de parameter λ geïntroduceerd wordt.

Opmerking: Wanneer het linkerlid van (1) minstens een tweeterm is, komt minstens een der x_i zowel zonder als met accent voor. Deze kan dan de onafhankelijk veranderlijke dienst doen.

Heeft men de onafhankelijk veranderlijke eenmaal gekozen dan vindt men voor elke keuze van $x_1, \dots, x_n$ weer een vergelijking van het type (1), ditmaal in $y_1, \dots, y_m$. De oplossingen hiervan verschijnen in de uiteindelijke uitdrukking van de y 's, vermenigvuldigd met de bij genoemde keuze behorende grootte $(x_1, \dots, x_n)_i$. Men kan deze vergelijking in de y 's op dezelfde wijze aanpakken als boven, waarbij men dan echter het bezwaar heeft b.v. de y_1 te vinden als functie van $x_1, \dots, x_n$ en laten we zeggen y_2 , waarbij de waarden van y_2 echter niet met alle waardencombinaties van $x_1, \dots, x_n$ verenigbaar zijn. Zo kan men b.v. de vergelijking

$$x y_1 y_2' + x y_1' y_2 + x' y_1 y_2 + x' y_1' y_2$$

oplossen in de vorm

$$y_1 = y_2' + \lambda x'$$

$$y_2 = \mu x + x'$$

waarbij dus de combinatie $x = 0, y_2 = 0$ uitgesloten is. Een betere oplossing is $y_1 = \mu' x + \lambda x'$

$$y_2 = \mu x + x',$$

alhoewel men in dit geval waarschijnlijk de parameterloze oplossing $y_2 = x y_1' + x' y_1 + x' y_1'$ zou verkiezen.

Een iets andere methode waarbij men zich niet de vraag behoeft te stellen welke veranderlijke men als onafhankelijk wil beschouwen, is de volgende.

We schrijven het linkerlid van (1) in de gedaante:

$$x_1 f_1(x_2, \dots, x_n) + x_1' g_1(x_2, \dots, x_n) = 1.$$

Dit vereist allereerst dat $f_1 = 1$ of $g_1 = 1$ of beiden, dus

$$f_1 g_1' + f_1' g_1 + f_1 g_1 = 1$$

en tevens

$$x_1 = f_1 g_1' + \alpha_1 f_1 g_1.$$

Is toevallig $f_1' = g_1'$, dan is hiermede het vraagstuk opgelost:

$x_2, \dots, x_n$ zijn dan onafhankelijk veranderlijk en $x_1 = f_1(x_2, \dots, x_n)$.

Zo niet, dan gaan we op dezelfde wijze door met

$$f_1 g_1' + f_1' g_1 + f_1 g_1 = x_2 f_2(x_3, \dots, x_n) + x_2' g_2(x_3, \dots, x_n) = 1$$

enz. tot we komen aan

$$x_n f_n + x_n' g_n = 1 \quad (f_n \text{ en } g_n \text{ bevatten geen } x \text{ meer})$$

$$\left. \begin{array}{l} \underline{a.} \quad f_n = 0 \\ \quad \quad g_n = 1 \end{array} \right\} \quad \text{of} \quad \left. \begin{array}{l} f_n = 1 \\ \quad \quad g_n = 0 \end{array} \right\}$$

We vinden voor x_n de waarden 0 resp. 1.

b. $f_n = 1, g_n = 1, x_n$ willekeurig, d.w.z. $x_n = \alpha_n$.

c. $f_n = 0, g_n = 0$. Het stelsel bezit geen oplossing.

Geval c kunnen we gevoegelijk uitsluiten als het linkerlid van (1)

minstens een eenterm is, daar we dan onmiddellijk een oplossing kunnen aangeven. In de gevallen a en b vinden we door terug te werken de volledige oplossing.

Voorbeeld: $x y z + x y' z' + x' y' z = 1$

$$x(y z + y' z') + x' y' z = 1$$

$$x = y z + y' z'$$

$$y z + y' z' + y' z = 1$$

$$y z + y'(z+z') = 1$$

$$y = \alpha z$$

$$z + z' = 1$$

$$z = \beta \text{ of}$$

$$x = \alpha\beta + \beta'$$

$$y = \alpha\beta$$

$$z = \beta.$$

We beschouwen nu de volgende transformatie

$$y_i = \sum_{j=0}^{2^n-1} a_{ij}(x_1, \dots, x_n)_j \quad (i = 1, \dots, n) \quad (4)$$

Hierbij geeft het woord "volledig" aan, dat voor de 2^n verschillende waardencombinaties van $x_1, \dots, x_n$ de y 's ook 2^n verschillende waardencombinaties doorlopen, hetgeen het geval is als de kolommen van de matrix $||a_{ij}||$ (n rijen, 2^n kolommen) alle getallen 0 t/m 2^n-1 representeren.

We vragen nu naar de inverse transformatie en zullen te dien einde $x_1, \dots, x_n$ uit bovenstaande vergelijkingen gaan oplossen.

Uit (4) leiden we af:

$$\left. \begin{aligned} (x_1, \dots, x_n)_j a_{ij} y'_i &= 0 \\ (x_1, \dots, x_n)_j a'_{ij} y_i &= 0 \end{aligned} \right\} \begin{aligned} i &= 1, \dots, n \\ j &= 0, \dots, 2^n-1 \end{aligned} \quad (5)$$

Een bepaalde term $(x_1, \dots, x_n)_j (y_1, \dots, y_n)_k$ komt in de linkerleden alleen dan niet voor als $(a_{1j}, \dots, a_{nj})_k = 1$, d.w.z. we mogen (5) vervangen door

$$(x_1, \dots, x_n)_j (y_1, \dots, y_n)_k (a_{1j}, \dots, a_{nj})'_k = 0.$$

$$j = 0, \dots, 2^{n-1}$$

$$k = 0, \dots, 2^{n-1}.$$

Of ook

$$\sum_{j=0}^{2^{n-1}} \sum_{k=0}^{2^{n-1}} (x_1, \dots, x_n)_j (y_1, \dots, y_n)_k (a_{1j}, \dots, a_{nj})'_k = 1. \quad (6)$$

Aangezien uit onze afspraak volgt dat voor bepaalde k $(a_{1j}, \dots, a_{nj})'_k = 1$ voor een en slechts een waarde van j geldt, staan hier precies 2^n termen zoals te verwachten was.

Uitpellen van (6) levert ons tenslotte:

$$x_n \sum_{k=0}^{2^{n-1}} (y_1, \dots, y_n)_k \sum_{l=0}^{2^{n-1}-1} (a_{1,2l+1}, \dots, a_{n,2l+1})'_k$$

$$+ x'_n \sum_{k=0}^{2^{n-1}} (y_1, \dots, y_n)_k \sum_{l=0}^{2^{n-1}-1} (a_{1,2l}, \dots, a_{n,2l})'_k = 1 \text{ of}$$

$$x_n = \sum_{k=0}^{2^{n-1}} (y_1, \dots, y_n)_k \sum_{l=0}^{2^{n-1}-1} (a_{1,2l+1}, \dots, a_{n,2l+1})'_k$$

$$\text{Algemeen. } x_{n-m} = \sum_{k=0}^{2^{n-1}} (y_1, \dots, y_n)_k \sum_{j=0}^{2^{n-1}} \left\{ \left[\frac{j}{2^{m+1}} + \frac{1}{2} \right] - \left[\frac{j}{2^{m+1}} \right] \right\} \times$$

$$\times (a_{1j}, \dots, a_{nj})'_k.$$

Voor de inverse matrix $||b_{ij}||$ gedefiniëerd door

$$x_i = \sum_j b_{ij} (y_1, \dots, y_n)_j \quad \begin{matrix} i = 1, \dots, n \\ j = 0, \dots, 2^{n-1} \end{matrix}$$

geldt dus nu

$$b_{ij} = \sum_{k=0}^{2^{n-1}} \left\{ \left[\frac{k2^{i-1}}{2^n} + \frac{1}{2} \right] - \left[\frac{k2^{i-1}}{2^n} \right] \right\} (a_{1k}, \dots, a_{nk})'_j.$$

Als voorbeeld vragen we naar de omkering van de transformatie

$$y_1 = x_1 x_2' + x_1' x_2$$

$$y_2 = x_2.$$

Toepassing van de formule levert

$$x_1 = y_1 y_2' + y_1' y_2$$

$$x_2 = y_2.$$

Deze heeft dus dezelfde gedaante als de oorspronkelijke transformatie en is mitsdien een van de, wat we zouden kunnen noemen, vierkantswortels uit de eenheidstransformatie.

Als laatste voorbeeld beschouwen we de speciale telcode, die we hier voor 3 lijnen nog even opschrijven:

d_2	d_1	d_0	Aanvaarden we als definitie voor deze code dat
0	0	0	d_0 begint met 1 nul gevolgd door 2 enen, 2
0	0	1	nullen enz. i.h.a. dat d_k begint met 2^k nullen
0	1	1	gevolgd door 2^{k+1} enen, 2^{k+1} nullen enz., dan
0	1	0	kunnen we voor het element d_{kj} ($k = 0, \dots, n-1$,
1	1	0	$j = 0, \dots, 2^n-1$) de volgende uitdrukking op-
1	1	1	schrijven.
1	0	1	
1	0	0	

$$d_{kj} = \left[\frac{j+3 \cdot 2^k}{2^{k+2}} \right] - \left[\frac{j+2^k}{2^{k+2}} \right].$$

Schrijven we het getal j als $e_{n-1,j} 2^{n-1} + e_{n-2,j} 2^{n-2} + \dots + e_{1,j} 2 + e_{0,j}$ dan krijgen we, na in beide gehelensymbolen datgene geschrapt te hebben wat zeker geheel is en tegen elkaar wegvalt:

$$d_{kj} = \left[\frac{e_{0,j} + e_{1,j} 2 + \dots + e_{k+1,j} 2^{k+1} + 32^k}{2^{k+2}} \right] - \left[\frac{e_{0,j} + e_{1,j} 2 + \dots + e_{k+1,j} 2^{k+1} + 2^k}{2^{k+2}} \right].$$

Gemakkelijk ziet men in dat het linker gehelensymbool de uitkomst 1 geeft wanneer $e_{k+1,j} = 1$ of $e_{k,j} = 1$ of beiden, terwijl het rechter alleen dan een uitkomst 1 levert, als $e_{k+1,j} = e_{k,j} = 1$. Derhalve

$$d_{kj} = (e_{kj} e'_{k+1,j} + e'_{kj} e_{k+1,j} + e_{k,j} e_{k+1,j}) - e_{kj} e_{k+1,j} = e_{kj} e'_{k+1,j} + e'_{kj} e_{k+1,j}.$$

Bedenkende dat voor $k = n-1$ deze betrekking zich vereenvoudigt tot $d_{n-1,j} = e_{n-1,j}$ en de index j verder weglatend, vinden we dus

$$\begin{aligned} d_{n-1} &= e_{n-1} \\ d_{n-2} &= e'_{n-1} e_{n-2} + e_{n-1} e'_{n-2} \\ d_{n-3} &= e'_{n-2} e_{n-3} + e_{n-2} e'_{n-3} \\ &\vdots \\ d_0 &= e'_1 e_0 + e_1 e'_0. \end{aligned} \tag{7}$$

Dit zijn dus de transformatieformules die de digits van onze code uitdrukken in die van de binaire code.

Oplossing van (7) geeft nu

$$\begin{aligned} e_{n-1} &= d_{n-1} \\ e_{n-2} &= e'_{n-1} d_{n-2} + e_{n-1} d'_{n-2} \\ e_{n-3} &= e'_{n-2} d_{n-3} + e_{n-2} d'_{n-3} \\ &\vdots \\ e_0 &= e'_1 d_0 + e_1 d'_0 \end{aligned} \tag{8a}$$

of

$$\begin{aligned} e_{n-1} &= \sigma(d_{n-1}) \\ e_{n-2} &= \sigma(d_{n-1}, d_{n-2}) \\ &\vdots \\ e_0 &= \sigma(d_{n-1}, d_{n-2}, \dots, d_0) \end{aligned} \tag{8b}$$

waarbij $\sigma(x_1, \dots, x_m)$ betekent de som van al die termen $(x_1, \dots, x_m)_j$, waarin een oneven aantal der x_i 's geen accent hebben, d.w.z. het somcijfer van $x_1, \dots, x_m$. Uitdrukking 8a zal wellicht uit technisch oogpunt de voorkeur verdienen, terwijl voor onze verdere beschouwingen 8b van belang zal blijken. We zien hieruit n.l. dat we door wijziging van d_m de digits $e_0, \dots, e_m$ alle wijzigen en vragen ons nu af of het mogelijk is de code te doorlopen in elk gewenst even aantal stappen $\leq 2^n$ met behoud van de eigenschap dat bij elke stap slechts een digit wijzigt (dat doorlopen in een oneven aantal stappen niet mogelijk is, is direct in te zien). We zullen trachten dit zo te doen dat we slechts een maal een sprong maken groot $2m+1$ ($2m$ overslaan) en de vraag is dus of dit voor alle $m \leq 2^{n-1}-1$ mogelijk is en zo ja, uitgaande van welk rijnummer. Willen we b.v. in 13 lijnen een sprong 0100111010001 maken dan schrijven we deze als 0111111111111, waarbij de regel dus is, dat we een groep van p nullen gevolgd door een 1 vervangen door een 1, gevolgd door p keer 1. ') We gaan nu uit van het rijnummer 0001100010111 dat dus enen heeft op de plaatsen, waar in het voorgaande getal 1 stond, wijzigen nu d_{11} waardoor $e_{11}, \dots, e_0$ alle veranderen en komen zo terecht op 0110011101000, waarmee we de gewenste sprong hebben gemaakt.

Algemeen. willen we een sprong maken, groot

$$2^{p_1} + 2^{p_2} + \dots + 2^{p_q} + 1 \quad (p_1 > p_2 > \dots > p_q > 0) \\ p_1 < n-1$$

dan schrijven we deze als

$$2^{p_1} + 2^{p_1-1} - 2^{p_2} (2^{p_1-p_2-1} - 1) + 2^{p_2-1} - 2^{p_3} (2^{p_2-p_3-1} - 1) + \dots \\ \dots + 2^{p_q-1} - (2^{p_q-1} - 1).$$

') Uitgezonderd de meest significante groep.

We gaan nu uit van het rijnummer $2^{p_2} (2^{p_1 - p_2 - 1} - 1) + 2^{p_3} (2^{p_2 - p_3 - 1} - 1) + \dots$

$$\dots + (2^{p_q - 1} - 1) = 2^{p_1 - 1} - 2^{p_2} + 2^{p_2 - 1} - 2^{p_3} + \dots + 2^{p_q - 1} + 2^{p_q - 1} - 1,$$

wijzigen hiervan de digits $e_{p_1} e_{p_1 - 1} \dots e_0$ door wijziging van d_{p_1} en

komen zo terecht op

$$2^{p_1} + 2^{p_1 - 1} + 2^{p_2 - 1} + \dots + 2^{p_q - 1}.$$

We blijken dus uit te kunnen gaan van een rijnummer, waarvan geldt

$$e_{p_1} = e_{p_1 - 1} = 0 \text{ dus zeker } e_{n-1} = e_{n-2} = 0 \text{ (eerste kwartgedeelte),}$$

terwijl het rijnummer waar we op terecht komen, groter is dan waar we van uitgingen, zodat we met onze sprong niet over het eind van onze code heenschieten.

Voorbeeld. $n = 4$, sprong $11 = 1011$.

$$p_1 = 3$$

$$p_2 = 1$$

We gaan dus nu uit van rij $2^2 - 2^1 + 2^0 - 1 = 2$, wijzigen d_3 en komen terecht op $2^3 + 2^2 + 2^0 = 13$, hetgeen klopt.

CONTROLE OP BINAAL-DECIMALE CONVERSIE

E.W. Dijkstra

Lezing in het colloquium Moderne Rekenmachines van het Mathematisch Centrum op 30 januari 1954 te Amsterdam.

Samenvatting:

Besproken worden de algemene eigenschappen van de type-routines voor de ARRA, waarbij gestreefd is naar:

- 1) minimale kans, dat een incidentele fout, tijdens het uittypen optreden, onopgemerkt blijft,
- 2) maximale kans, dat, als een fout gesignaleerd wordt, deze inderdaad in het essentiële, en niet in het accessoire gedeelte der berekening is geweest,
- 3) minimaal tijdsverlies (d.w.z. zo min mogelijk vermenigvuldigingen en delingen, die vijf maal de duur van de normale opdracht hebben),
- 4) uniforme faciliteiten ten aanzien van de wijze, waarop de subroutines in het programma worden ingelast.

Aangegeven worden de methodes, waarop bovengestelde doeleinden bereikt worden; hierbij wordt de aard van analyse, waar deze problemen aan onderworpen dienen te worden, voldoende geïllustreerd.

Van de volgende gegevens omtrent de ARRA, en de volgende notaties zal gebruik worden gemaakt.

Registers - het rekenkundig orgaan kent er twee, nl. A en S - en adressen in het geheugen hebben de uniforme capaciteit van 29 binalen en een tekencijfer; met "het laagste cijfer" wordt bedoeld het cijfer dat beantwoordt aan de laagste macht van 2. Het tekencijfer heet het hoogste cijfer.

De inhoud van een register of adres wordt aangeduid met (A), (S) of (n). Het positieve of negatieve gehele getal, gevormd door teken-

cijfer en 29 binalen in adres n wordt aangeduid met $[n]$
 $(-2^{29} < [n] < +2^{29})$. Interpreteert men de komma tussen het tekencijfer
 en de op een na hoogste binaal, dan noteert men dit met $\{n\} = [n] \cdot 2^{-29}$.
 2^{-29} wordt genoteerd als p (penten!).

Bij de vermenigvuldiging worden de 58 binalen van het product ge-
 lijkelijk over A en S verdeeld, d.w.z.: in elk het tekencijfer gelijk
 aan het teken van het product, de hoogste 29 binalen van het product
 komen in volgorde in de resterende plaatsen van A , de laagste 29 in S .

Inhouden voor de operatie met een enkel, na de operatie met een
 dubbel accent aangevend, kunnen de twee vermenigvuldigingen, waartoe
 de ARRA in staat is, beschreven worden:

$$\begin{aligned} 16/n \quad [A'] + [n] \cdot [S'] &= 2^{29} [A''] + [S''] \text{ (additieve verm.)} \\ 18/n \quad [n] \cdot [S'] &= 2^{29} [A''] + [S''] \text{ (schone verm.)} \end{aligned}$$

Delen we deze betrekking door 2^{29} , dan luidt de omschrijving:

$$\begin{aligned} 16/n \quad \{A'\} + [n] \cdot \{S'\} &= [A''] + \{S''\} \\ 18/n \quad [n] \cdot \{S'\} &= [A''] + \{S''\}. \end{aligned}$$

Het verdient aanbeveling het proces der binaal-decimale conversie
 te vergelijken met het inverse, dat bij de invoer zulk een grote rol
 speelt. De asymmetrie tussen deze twee processen komt niet zozeer voort
 uit het feit, dat dan van eigen naar vreemd, dan van vreemd naar eigen
 talstelsel moet worden geconverteerd, als wel uit het feit, dat de bij
 de invoer te bewerken informatie in serie binnenkomt, bij het typen
 de af te voeren resultaten (de opeenvolgende cijfers!) in serie dienen
 geproduceerd te worden. Dit volgt al uit het feit, dat zoals later zal
 blijken, we de gebeurtenissen in de machine, ook tijdens het conversie-
 proces, adequaat 10-tallig kunnen beschrijven.

Invoer

Het conversieprogramma van het 10-tallig naar het 2-tallig stelsel
 tijdens de invoer bouwt gehele getallen op, gebruikmakend van de zg.
 "snelle additieve vermenigvuldiging met 10": opdracht 24/16 (in functie
 identiek aan 16/n, mits $[n] = 10$), en wordt slechts gebruikt, zolang deze
 vermenigvuldiging $[A''] = 0$ aflevert (antwoord dus steeds kleiner dan 2^{29}).

Dan luidt de omschrijving: $[A'] + 10[S'] = [S'']$.

De bandleesopdracht plaatst de successievelijke decimale cijfers met de hoogste te beginnen, in $[A]$, onderbroken door boven omschreven additieve vermenigvuldiging met 10, terwijl aanvankelijk $[S] = 0$ wordt gemaakt. Wordt dus 123 van de band gelezen, dan is $[S]$ in de successievelijke stadia

$$\begin{aligned}[S] &= &= 0 & \quad (\text{i.e. nog geen cijfer ingelezen!}) \\[S] &= 0.10 + 1 = 1 \\[S] &= 1.10 + 2 = 12 \\[S] &= 12.10 + 3 = 123.\end{aligned}$$

Wanneer op de band het einde van het getal geïndiceerd wordt, is dat in een cyclus te programmeren, zodat men getallen van een variabel aantal cijfers in kan lezen.

Breuken worden ingebracht door het 10^8 -voud als geheel getal in te lezen, en dit vervolgens door 10^8 te delen.

$20/n$, de deelopdracht deelt (mits teken (A) = teken (S)):

$$2^{29}[A'] + [S'] \text{ door } [n], [A''] = \text{rest}, [S''] = \text{quotiënt achterlatend.}$$

Deling door 2^{29} geeft als beschrijving:

$$[A'] + \{S'\} \text{ gedeeld door } [n], \{A''\} = \text{rest}, \{S''\} = \text{quotiënt achterlatend.}$$

(Aan de voorwaarde $|[A']| < |[n]|$ moet voldaan zijn).

Volgens de laatste beschrijving kan men het inlezen van breuken voltooien door een deling.

Uitvoer

Leest men primair een geheel getal, bij het typen typt men primair een breuk. Men kan hier voor $24/17$, het "schoon" analogon van $24/16$, het "snel maal 10"-analogon van $18/n$, gebruiken; de omschrijving luidt dan:

$$24/17 \quad 10\{S'\} = [A] + \{S''\}.$$

De type-opdracht typt "de laagste 4 binalen" van $[A]$ (de bandlees opdracht leest in de laagste 5 van $[A]$); als deze de getallen 0 t/m 9 vormen, worden de overeenkomstige cijfers getypt, 10 t/m 15 zijn voor punt, spatie, etc. gereserveerd.

Het eerste decimale cijfer achter de komma van een positieve breuk (uiteraard kleiner dan 1) is per definitie het gehele gedeelte van het 10-voud; voor het volgende cijfer geldt hetzelfde met betrekking tot het zojuist gevormde breukdeel. $24/17$ splitst juist het 10-voud in geheel gedeelte en breukdeel, nl. als $[A'']$ en $\{S''\}$. Door herhaling van $24/17$, gevolgd door een type-opdracht worden in eerste instantie de decimale cijfers van een breuk getypt. Tevens is het duidelijk, dat dit proces in een cyclus geprogrammeerd kan worden, mits men de telling in A, na de type-opdracht uitvoert.

Een geheel getal wordt getypt, door de breuk uit te rekenen, die bij uittyping de gewenste cijfers aflevert. Het typen van punt (eventueel door extra nul voorafgegaan) vervalt nu vanzelfsprekend.

Stel dat we het positieve getal G willen uittypen, als geheel getal van n cijfers. Om de breuk uit te rekenen, moeten we delen door 10^n ; omdat 10^8 de hoogste macht van 10 is, die in de registercapaciteit geborgen kan worden, kunnen zo gehele getallen van maximaal 8 cijfers worden uitgetypt.

Omdat deler een geheel getal, en quotiënt een breuk is, fungeert $[A'] + \{S'\}$ als deeltal. $[A'] = G$; omdat de deling over het algemeen een rest zal achterlaten, mag $\{S'\}$ niet gelijk aan nul worden gemaakt; in schier alle gevallen zou dan het gehele getal een eenheid te laag worden uitgetypt. M.a.w. het quotiënt moet naar boven afgerond worden. Daartoe wordt voor de deling $[S'] = 10^n$ gemaakt.

Men deelt dus: $G + 10^n$ door 10^n . Omdat de rest in $[A'']$ achtergelaten kleiner is dan 10^n (dus de feitelijke rest in $\{A''\}$ kleiner is dan 10^n) kan de bij deze deling verkregen breuk niet een te laag getal doen typen.

De breuk, na deling verkregen, is $g = G \cdot 10^{-n} + \partial p$, waar $0 < \partial \leq 1$; wordt nu door n successievelijke vermenigvuldigingen met 10 de breuk g met 10^n vermenigvuldigd, dan is na de laatste vermenigvuldiging met 10 het zg. residu in S gelijk aan $\partial p \cdot 10^n$. Aangezien dit te allen tijde kleiner is dan 1, heeft het steeds groter wordend residu tengevolge van de naar boven afgeronde deling door 10^n ontstaan, geen extra overdracht van het breuk gedeelte naar het gehele gedeelte tengevolge en wordt, behoudens rekenfouten, G exact uitgetypt. Het aldus ontstane residu is maximaal $p \cdot 10^8 = 0.186$, d.w.z. meer dan 80% van de spelling is nog onbenut gelaten. Dit wordt bij de controle op het uittypen gebruikt.

Een opdracht (nl. 24/12) leest het laatst getypte viertal binalen terug uit de schrijfmachine op de plaats waar ze vandaan gekomen zijn, dus op de vier laagste plaatsen van A . (In feite worden ze teruggelezen van een tussen register U : ze worden van zo ver mogelijk achterhaald). Men heeft dus door elke type-opdracht te laten volgen door de terugleesopdracht, voordat weer getypt wordt, de successievelijke tientallige cijfers weer tot zijn beschikking. Met behulp hiervan moet het te typen getal weer worden opgebouwd. Daartoe wordt elke type-opdracht gevolgd door de terugleesopdracht 24/12, en de nu volgende vermenigvuldiging met 10 wordt additief met 24/16 uitgevoerd (d.w.z. alle, behalve de eerste).

Hierdoor is na de n^{de} vermenigvuldiging met 10 het residu, vergeleken met dat, dat we schoon met 10 vermenigvuldigend gekregen zouden hebben (het schone residu), in eenheden p vermeerderd met het getal, dat gevormd wordt door de hoogste $n-1$ cijfers. Aangezien deze vermeerdering kleiner is dan 10^{n-1} , is het totale residu kleiner dan $(10^n + 10^{n-1})p$, dat onder alle omstandigheden kleiner is dan 1 en dan geen extra overdracht naar het getypte gehele gedeelte geven kan.

Ons rest de opgave, om deze vermeerdering van het "schone residu" te scheiden. Echter: als het schone residu vermeerderd wordt met de rest, die na de aanvankelijke deling door 10^n overblijft, wordt $[S] = 10^n$. Door dus deze rest bij ons nu verkregen resultaat op te tellen, en daarna $[S]$ met 10^n te verminderen, wordt $[S] = \text{gehele}$

getal, gevormd door de hoogste $n-1$ decimale cijfers van het getypte getal. Teruglezen van het laatste decimale cijfer en een additieve vermenigvuldiging met 10 bouwen in S een resultaat op, dat nauwelijks gelijk kan zijn aan het oorspronkelijke te typen getal, indien een type- of de conversiefout is ingeslopen. Met dit resultaat in S stuurt de subroutine de besturing terug in het hoofdprogramma.

Omdat schrijven uit een register naar een adres de inhoud van het register bederven kan, kan een controle op een registerinhoud er zich slechts over uitspreken, of het bedoelde getal goed in A of S stond, niet of het daarna nog goed in het register staat. Dit impliceert, dat slechts over de inhoud van adressen een zekere garantie kan worden uitgesproken (lezen uit een adres wordt geacht de inhoud onveranderd te laten). Het is echter ongewenst de routines de inhoud van een bepaald adres uit te laten typen. De type-routines krijgen alle het te typen getal in S mee. Maar dan moet ter controle het te typen getal ten tweeden male uit de (variabele!) adresplaats worden aangehaald. Dit moet dus in het hoofdprogramma geschieden: de type-routines laten uniform als alles goed gegaan is bij terugkomst in het hoofdprogramma in S het getal achter, dat ze bij binnenkomst meekregen. Dan wordt het te typen getal wederom uit het geheugen aangehaald, doordat het van S wordt afgetrokken.

Het al of niet nul zijn van S is dan het criterium of het gewenste of een ander getal, goed of fout getypt is.

Bij het inlassen van een controle op een breuk, treedt de moeilijkheid op, dat de getypte decimale cijfers de binale representatie niet eenduidig bepalen. Wat de typeroutines na afloop in S afleveren, hangt dus, behalve van de getypte cijfers, tevens van de oorspronkelijke breuk af, en is gelijk aan deze, als geen fout is opgetreden. De benodigde operaties gaven en passant de mogelijkheid exact op n decimalen achter de komma af te ronden. De te typen cijfers vormen als geheel getal beschouwd het gehele gedeelte van het op eenheden afgeronde 10^n -voud van de breuk. (Omdat de eenheden in alle talstelsels "samenvallen" kan hier exact afgerond worden!). Dit gehele getal zou men, voorafgegaan door een punt, kunnen uittypen als boven beschreven.

Men zou beginnen met een deling, die haast hetzelfde resultaat aflevert, als de oorspronkelijke breuk! Dit was de overweging, die leidde tot de poging, deze deling te laten vervallen.

De gevonden methode is als volgt:

Na vermenigvuldiging met 10^n moet naar boven afgerond worden, m.a.w. dient voor een extra overdracht gezorgd te worden, als $\{S\} \geq \frac{1}{2}$, of $[S] \geq 2^{28}$. Het verschil tussen deze mogelijkheden wordt uitsluitend bepaald door het op een na hoogste cijfer van S , dat we met a aanduiden. ($a = 0$, geen extra overdracht, $a = 1$, wel extra overdracht).

Als nu de oorspronkelijke breuk, die in eenheden p geborgen staat, wordt voorgesteld door fp (f geheel), dan maakt men

$$\{S\} = fp + ap + [2^{28} \cdot 10^{-n}]p \quad [] \text{ betekent hier: } \\ \text{"het gehele gedeelte van".}$$

Beschouwt men het primaire residu, als n maal met 10 vermenigvuldigd is:

$$(r + 10^n a + 10^n [2^{28} \cdot 10^{-n}])p$$

als rp gelijk is aan het breukgedeelte van $10^n fp$.

Als $r < 2^{28}$, dan is $a = 0$; $10^n [2^{28} \cdot 10^{-n}] < 2^{28}$.

Het residu is kleiner dan $(2^{28} + 0 + 2^{28}) \cdot p = 1$, en geen extra overdracht heeft plaats gevonden; zulks was de bedoeling.

Als $r > 2^{28}$, dan is $a = 1$,
dan geldt:

$$10^n a + 10^n [2^{28} \cdot 10^{-n}] = 10^n (1 + [2^{28} \cdot 10^{-n}]) > 2^{28}, \text{ omdat}$$

tussen de haken een naar boven afgerond quotiënt staat.

Het residu is groter dan $(2^{28} + 2^{28})p = 1$, en wel een extra overdracht heeft plaats gevonden (en ook niet meer dan 1); zulks was de bedoeling.

Deze overdracht vermindert onze uitdrukking voor het primaire residu met $1 = 2^{29}p$ tot het feitelijke, schone residu:

$$(r + (10^n - 2^{29})a + 10^n [2^{28} \cdot 10^{-n}]).$$

De bewering is dat we nu vrijelijk de vermenigvuldigingen additief kunnen uitvoeren: Noem het gehele getal, gevormd door de eerste $(n-1)$ decimale cijfers x . Dan vormt men

$$(r + (10^n - 2^{29})a + 10^n [2^{28} \cdot 10^{-n}] + x)p \text{ als additief}$$

residu:

het is echter makkelijk om aan te tonen, dat de toevoeging van x nu nooit een extra overdracht geven kan. Omdat x slechts $n-1$ decimale cijfers beslaat, en in de decimale representatie van $2^{28} = 268435456$ geen nul voorkomt is

$$10^n [2^{28} \cdot 10^{-n}] + x < 2^{28}.$$

Hetzelfde geldt voor $r + (10^n - 2^{29})a$.

(als $r < 2^{28}$ is $a = 0$; als $r > 2^{28}$ dan geldt:

$$10^n - 2^{28} = 2^{28} + (10^n - 2^{29}) \leq r + (10^n - 2^{29}) \leq (2^{29} - 1) + (10^n - 2^{29}) = 10^n - 1.$$

De scheiding x en het residu geschiedt door een tekentest, nl. om te bepalen of $a = 0$ of $a = 1$.

Vermindert men het residu met rp , dan is het restant

positief, als $a = 0$, en negatief, als $a = 1$.

In het laatste geval vermindert men S met $10^n - (2^{29} - 1)$ en vermeerderd de inmiddels teruggelezen laatste decimaal met 9; in het eerste geval laat men deze onveranderd.

In beide gevallen vermindert men S met $10^n [2^{28} \cdot 10^{-n}]$;

Als de laagste decimaal y is, is:

als $a = 0$

als $a = 1$

$$[A] = y \quad [S] = x$$

$$[A] = y+9 \quad [S] = x + (10^n - 2^{29}) - (10^n - 2^{29} + 1) = x-1.$$

additieve vermenigvuldiging levert

$$10x + y$$

$$10x + 10y + 9 = 10x + (y-1).$$

Het resultaat is dus $10x + y - a$, de oorspronkelijke inhoud van A ,

na de eerste vermenigvuldiging met 10^n : Plaatst men dit in $[A]$, en
rp in $\{S\}$, dan moet een nu opgaande deling door 10^n in S het oorspron-
kelijke resultaat opleveren.

DYNAMICA VAN REKENMACHINES

A. van Wijngaarden

Lezing in het colloquium Moderne Rekenmachines van het Mathematisch Centrum op 27 maart 1954 te Amsterdam.

Wij houden ons hier bezig met het gedrag van eenvoudige rekenmachines, bestaande uit een doosje met een aantal invoerlijnen en een aantal uitvoerlijnen. Alles is digitaal en op een bepaald tijdstip voeren we op de invoerlijnen cijfers $u_1, u_2, \dots, u_k$ in het m -tallig stelsel binnen, tengevolge waarvan op de uitvoerlijnen cijfers $v_1, v_2, \dots, v_l$ verschijnen. Korteheidshalve kunnen we spreken van een vector $\underline{u}$ en een vector $\underline{v}$. Evenwel hangt $\underline{v}$ niet alleen van $\underline{u}$ af, maar ook van de toestand waarin het doosje zich bevindt. Er zullen nl. i.h.a. geheugenelementen aanwezig zijn binnen de doos en het zal van de stand daarvan ook afhangen wat er gebeurt. Trouwens ook deze stand verandert i.h.a. bij iedere stap van het "rekenproces". De toestand van de geheugenelementen is ook weer door cijfers voor te stellen en deze cijfers op voorgeschreven wijze aan elkaar verbonden definiëren een getal, de "toestand" van de doos genoemd. Als men de constructie van de doos kent en deze toestand, dan kan men precies nagaan, hoe de doos op een vector $\underline{u}$ reageert, d.w.z. welke vector $\underline{v}$ ze aflevert en in welke nieuwe toestand de doos achterblijft. Het is overigens niet gezegd, dat men de toestand van alle geheugenelementen behoeft te kennen, want het kan zijn, dat bepaalde elementen logisch overvloedig zijn, of algemener althans, dat slechts een bepaalde functie van alle elementaire toestanden karakteristiek is voor het gedrag van de doos. Deze toestand noemen we A (een getal dus). Het blijkt nu, dat men in bepaalde gevallen een soort dynamica kan opstellen, waaraan Duparc en van Wijngaarden hebben gewerkt. Enkele aspecten

hiervan zullen wij hier behandelen, waarbij geenszins gestreefd is naar algemeenheid, maar slechts zoveel zal beschouwd worden als nodig is om karakteristieke punten naar voren te halen en enkele praktische resultaten te bereiken.

Noemen we A de toestand voor en A^* die na de rekenstap, dan blijken er allereerst een aantal elementaire doosjes te zijn, die voldoen aan de volgende vergelijking:

$$mA^* - A = \underline{P} \underline{u} - \underline{Q} \underline{v}. \quad (1)$$

Hierin zijn $\underline{P}$ resp. $\underline{Q}$ constante vectoren met k resp. l elementen, die karakteristiek zijn voor de opbouw van het doosje. De bedoeling is dus dat

$$\begin{aligned} \underline{P} \underline{u} &= P_1 u_1 + P_2 u_2 + \dots + P_k u_k, \\ \underline{Q} \underline{v} &= Q_1 v_1 + Q_2 v_2 + \dots + Q_l v_l, \end{aligned} \quad (2)$$

waarin P_j en Q_j zekere (gehele) getallen zijn. Is k of l gelijk aan 1, dan laten wij de index j weg.

Voorbeelden zijn:

1e. De doorvoer (E); $A = 0$, $P = Q$ (bijv. $= 1$).

2e. De invertor (I); $u = m-1-v$, $A = 1$, $P = 1$, $Q = -1$.

3e. De verdragingslijn (S); $u_{n-s} = v_n$, $A = S$, $P = m^S$, $Q = 1$.

Hierin is S het getal van s cijfers, dat zich in de verdragingslijn bevindt, de minst significante zijde aan de uitgangszijde gerekend. De index bij u en v slaat hier op het nummer van de stap.

4e. De opteller (C); $u_1 + u_2 + C = v + m C^*$,

$$A = C, P_1 = P_2 = Q = 1.$$

Hierin is C de overdracht (carry), die zich in de opteller bevindt van de vooraafgaande stap.

In het algemeen zullen wij een doos die aan (1) voldoet voorstellen door het symbool $B_{k,l}$. Men kan nu ingewikkelder dozen $B_{k,l}$ maken, door elementen als bovengenoemd in serie of parallel te schakelen. Daarvoor zijn eenvoudige schakelregels, als bijv.

Serieschakeling van een $B'_{k,1}$ en een $B''_{1,1}$ levert een $B_{k,1}$, en wel is:
 $A = P''A' + Q'A'', P_j = P''P'_j, Q = Q'Q''.$

Parallelschakeling van een $B'_{1,1}$ en een $B''_{1,1}$ via een opteller C levert een $B_{2,1}$ en wel is:

$$A = Q''A' + Q'A'' + Q'Q''C, P_1 = P'Q'', P_2 = P''Q', Q = Q'Q''.$$

Door deze en soortgelijke regels bij herhaling toe te passen kan men ingewikkelde schakelingen eenvoudig doorrekenen.

Een eenvoudig voorbeeld is:

$$\begin{array}{ccccccc} u_1 & \rightarrow & (E) & \rightarrow & & & \\ & & & & (C) & \rightarrow & (I) \rightarrow v \equiv u_1 \rightarrow (B) \rightarrow v \\ u_2 & \rightarrow & (S) & \rightarrow & & & \\ & & & & & & u_2 \rightarrow \end{array}$$

Deze doos (B) kunnen wij nu in twee toestanden schakelen.

Toestand 1: (B'): $u_1 = u_2,$

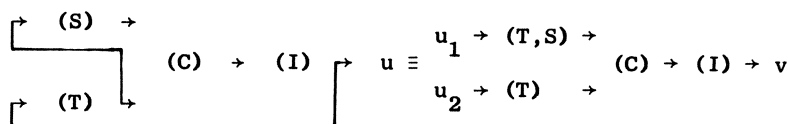
Toestand 2: (B''): $u_2 = v,$

en deze kunnen weer in serie geschakeld worden. Wij vinden dan voor de karakteristieke grootheden:

	(E)	(S)	$\begin{smallmatrix} (E) \\ (S) \end{smallmatrix} (C)$	(I)	$\begin{smallmatrix} (E) \\ (S) \end{smallmatrix} (C) (I)$	(B')	(B'')	(B')(B'')
A	0	S	S+C	1	S+C+1	S'+C'+1	S''+C''+1	S'-S''+C'-C''
P ₁	1	m ^S	1	1	1	m ^S +1	1	m ^S +1
P ₂			m ^S		m ^S			
Q	1	1	1	-1	-1	-1	-(m ^S +1)	m ^S +1

Men ziet, dat als men begint met een instelling $S' = S'', C' = C''$, bijv. beide dozen gevuld met louter nullen, dat $A = 0$, en daar voorts $P = Q$ is de complete schakeling weer een doorvoer. De dozen (B') en (B'') zijn dus elkaars reciproke, zij vormen een codator en een decodator.

Een ander voorbeeld is:



De equivalentie geldt, mits in beide (T)'s hetzelfde getal T oorspronkelijk geborgen wordt geacht, en $u = u_1 = u_2 = v$ achteraf wordt gesteld. Men vindt:

	(T+S)	(T)	(T+S) (T) (C)	(I)	(T+S) (T) (C) (I)
A	$Tm^s + S$	T	$T(m^s + 1) + S + C$	1	$T(m^s + 1) + S + C + 1$
P_1	m^{s+t}	m^t	m^{s+t}	1	m^{s+t}
P_2			m^t		m^t
Q	1	1	1	-1	-1

Men stelt i.h.a. $P_1 + P_2 + \dots + P_k - Q_1 - Q_2 - \dots - Q_l = M$, waarin M de modulus van de doos wordt genoemd. Althans in het geval dat $l = 1$ is het mogelijk alle ingangen en uitgangen van de doos aan elkaar te verbinden. Het enige overblijvende cijfer noemen wij dan korthedshalve u en (1) neemt de vorm aan

$$mA^* - A = Mu. \quad (3)$$

Dergelijke schakelingen kunnen worden gebruikt voor het genereren van pseudo-willekeurige cijfers. Wij merken op, dat in bovenstaande schakeling

$$M = m^{s+t} + m^t + 1$$

en dat, zoals men gemakkelijk narekent voor de toestand A de grens geldt

$$1 \leq A \leq M-1.$$

Laat men de invertor weg, dan heeft men $M = m^{s+t} + m^t - 1$ en

$$0 \leq A \leq M.$$

Wij gaan nu over tot de behandeling van de vergelijking (2).

Voeren wij weer de index n als nummer van de stap in, dan luidt zij:

$$mA_{n+1} - A_n = Mu_n. \quad (4)$$

Voorts leggen wij de volgende beperkingen op:

$$0 \leq u_n \leq m-1, m > 1, M > 1, (m, M) = 1, \quad (5)$$

waarvan de eerste twee triviaal zijn en waarvan de derde en de vierde althans in de genoemde gevallen gelden. Tenslotte volgt uit (4), dat (A, M) een constante is, d.w.z. onafhankelijk van n . Wij stellen nog:

$$M' = \frac{M}{(A, M)}, \text{ dus } (A, M') = 1. \quad (6)$$

Geeft men A_n , dan volgen uit (4) zowel u_n als A_{n+1} . Immers, dan zijn bekend

$$\alpha_n \equiv A_n \pmod{m}$$

$$\mu \equiv M \pmod{m}$$

en krachtens (5) bestaat de reciproke ν van μ , zodat

$$\mu\nu \equiv 1 \pmod{m}.$$

Uit (4) volgt

$$-\alpha_n \equiv \mu u_n \pmod{m}$$

en dus is

$$-\nu\alpha_n \equiv u_n \pmod{m}$$

bekend en daarmee u_n , waaruit met A_n ook A_{n+1} volgt.

Omgekeerd volgen echter uit A_{n+1} niet zonder meer u_n en A_n . Wel volgt uit (4) direct $A_n \pmod{M}$, maar omdat wij geen grenzen voor de A_n hebben gegeven, zoals wel voor de u_n , volgt hieruit nog niet A_n zelf. Hier is evenwel wat op te vinden. Uit (4) volgt nl.

$$mA_1 - A_0 = M u_0$$

$$mA_2 - A_1 = M u_1$$

$$\dots\dots\dots$$

$$mA_n - A_{n-1} = M u_{n-1}$$

en hieruit

$$m^n A_n - A_0 = M(u_0 + m u_1 + \dots + m^{n-1} u_{n-1}).$$

Dus

$$\begin{aligned} 0 &\leq m^n A_n - A_0 \leq M(m^n - 1), \\ 0 &\leq A_n - m^{-n} A_0 \leq M(1 - m^{-n}), \\ m^{-n} A_0 &\leq A_n \leq M + m^{-n}(A_0 - M). \end{aligned}$$

Dus geldt als n voldoende groot is

$$0 \leq A_n \leq M. \quad (7)$$

Het is dus wel soms mogelijk, dat A heel groot positief of negatief is, maar na enige tijd wordt de A begrensd in het interval $0, M$. In de behandelde voorbeelden gold (7) trouwens al zonder meer. In ieder geval zullen wij pas beginnen te tellen als aan (7) is voldaan en een eventuele "beginspriet" (voor de kenners begrijpelijk) verwaarlozen we verder. Nu volgt direct uit (4) en (5), dat uit $A_n = 0$, volgt $A_{n+1} = 0$, uit $A_n = M$ volgt $A_{n+1} = M$ en uit $A_n \not\equiv 0 \pmod{M}$ volgt $A_{n+1} \not\equiv 0 \pmod{M}$ en omgekeerd. Er zijn dus eventueel twee triviale gevallen, nl. $A = 0$, waarin het apparaat altijd een 0 afvuurt en $A = M$, waarin het apparaat een $m-1$ afvuurt.

In het overblijvende, niet triviale geval geldt de verscherping van (7)

$$0 < A < M, \quad (8)$$

waaraan in het behandelde geval met invertor trouwens altijd is voldaan.

Nu weten wij dus, dat inderdaad uit A_{n+1} ook A_n en dus u_n volgen. Maar dan is de rij der getallen A_n zuiver periodiek. Immers A_n kan maar een eindig aantal waarden aannemen, dus in de rij $A_0, A_1, \dots$ moet een eerste element A_j zijn, dat gelijk is aan een voorgaand element A_{j-C_A} . Enerzijds volgt hieruit, dat nu ook $A_{j+1} = A_{j-C_A+1}$,

etc., maar ook, dat $j = C_A$. Immers anders zou volgen $A_{j-1} = A_{j-C_A-1}$ in strijd met de onderstelling dat A_j het eerst herhalende element zou zijn. C_A is dus de periode van A .

Uit het feit, dat A_n periodiek is volgt uit (4) direct, dat u_n periodiek is. Zij de periode van u gelijk aan C_u , dan is C_A een periode van u en dus

$$C_u | C_A. \quad (9)$$

Omgekeerd is

$$mA_{C_u+n+1} - A_{C_u+n} = Mu_{C_u+n} = Mu_n = mA_{n+1} - A_n,$$

dus

$$\begin{aligned} A_{C_u+n} - A_n &= m(A_{C_u+n+1} - A_{n+1}) \\ &= m^2(A_{C_u+n+2} - A_{n+2}) \\ &= \dots \\ &= m^j(A_{C_u+n+j} - A_{n+j}), \end{aligned}$$

met willekeurig grote j . Dus bevat $A_{C_u+n} - A_n$ willekeurig veel factoren m , en omdat A_n eindig is, is dus $A_{C_u+n} = A_n$, m.a.w. C_u is een periode van A , dus

$$C_A | C_u. \quad (10)$$

Uit (9) en (10) volgt

$$C_A = C_u.$$

Om de periode van u te vinden, die ons voor een generator van pseudo-willekeurige cijfers interesseert, heeft men dus slechts de periode van A te bepalen en kan men de u buiten beschouwing laten. Dit gebeurt door (4) te beschouwen modulo M . Dan is:

$$\begin{aligned}
& m A_1 \equiv A_0 \pmod{M}, \\
& m A_2 \equiv A_1 \pmod{M}, \\
\text{dus} \quad & m^2 A_2 \equiv A_0 \pmod{M}, \\
\text{en i.h.a.} \quad & m^j A_j \equiv A_0 \pmod{M},
\end{aligned}$$

dus zeker met (6):

$$m^j A_j \equiv A_0 \pmod{M'}. \quad (11)$$

Kies nu $j = C_A$, dan is

$$m^{C_A} A_{C_A} \equiv m^{C_A} A_0 \equiv A_0 \pmod{M'},$$

dus

$$m^{C_A} \equiv 1 \pmod{M'}.$$

Is per definitie $C_m(M')$ de exponent van m modulo M' , d.w.z. de kleinste positieve exponent, waarvoor geldt

$$m^{C_m(M')} \equiv 1 \pmod{M'}.$$

dan volgt dus

$$C_m(M') \mid C_A. \quad (12)$$

Kies nu $j = C_m(M')$, dan is

$$m^{C_m(M')} A_{C_m(M')} \equiv A_{C_m(M')} \equiv A_0 \pmod{M'}.$$

Als $K = M/M' \neq 1$, kan men (4) en (7) en (8) alle door K delen en dus volgt uit (7) met discussie of (8) zonder meer, dat

$$A_{C_m(M')} = A_0,$$

m.a.w. $C_m(M')$ is een periode van A , of

$$C_A \mid C_m(M'). \quad (13)$$

Uit (12) en (13) volgt dus

$$C_A = C_m(M').$$

De exponent $C_m(M')$ is betrekkelijk gemakkelijk te berekenen met allerlei getallentheoretische trucjes, dus het vraagstuk van de periodebepaling is opgelost. Er schuilt echter nog de vervelende onzekerheid betreffende de grootte van K . Maar als men nu een doos kiest, zoals het behandelde geval met invertor, waarin (8) automatisch geldt en als men bovendien nog kan zorgen, dat M een priemgetal is, dan is noodzakelijk $K = 1$, hoe men ook start. Dus:

Als $0 < A < M$ en M is priem, dan $C_u = C_A = C_m(M)$.

In het behandelde geval met invertor moet men dus $M = m^{s+t} + m^t + 1$ priem zien te maken door passende keuze van s en t . Voorts is $s+t$ een maat voor de benodigde apparatuur. Voor $m = 2$ zijn vele priemen van dit type bekend. Een combinatie, waarbij de exponent relatief zeer hoog ligt is

$$m = 2, s = 10, t = 17, M = 2^{27} + 2^{17} + 1 = 134348801 \text{ (priem)}$$

en

$$C_u = C_2(M) = \frac{M-1}{2} = 67174400.$$

FROM RECURRING FRACTION TO RECURRING COMPUTING CIRCUIT.

H.J.A. Duparc

Lezing in het colloquium Moderne Rekenmachines van het Mathematisch Centrum op 24 april 1954 te Amsterdam.

0. In this report several periodic processes are studied from a general point of view. One of these periodic processes - already familiar to us since the early days of the primary school - is the theory of recurring fractions; another studied in a later stage of our mathematical life, is the theory of exponents of an integer modulo another integer. Both are aspects of the same kind of problems, viz. those on cyclic groups.

Here it is our intention to study also other processes which are necessarily periodic and can often be reduced to the above type; we mention the linear recurring sequences reduced modulo an integer and some other processes easily to be realised by simple computing circuits. Although the theory can be generalized further we treat only the here mentioned applications.

In all these processes besides the sequence $u_0, u_1, \dots$ of integers the periodicity of which is investigated also another sequence $v_0, v_1, \dots$ is introduced, the properties and period of which are closely connected to those of the original sequence.

1. Consider two sequences of integers

$$(u) \quad u_0, u_1, \dots$$

and

$$(v) \quad v_0, v_1, \dots,$$

satisfying the relations

$$(1) \quad U u_n = V v_n \quad 0 \leq u_n \leq m-1 \quad (n = 0, 1, \dots),$$

where m is a fixed given positive integer and where

$$U = U(E) = \sum_{h=0}^s p_h E^h \quad \text{and} \quad V = V(E) = \sum_{k=0}^t q_k E^k$$

are given polynomials with integer coefficients in the operator E which transforms any u_n into u_{n+1} and any v_n into v_{n+1} .

The operators U and V are subject to the following conditions

- I. $p_s = \pm 1$, $q_t = m$;
- II. $U(E)$ and $V(E)$ are relatively prime;
- III. $V(X)$ has no roots with absolute value ≥ 1 .

The condition I assures the possibility of determining u_n (if $n \geq s$) and v_n (if $n \geq t$) uniquely once the preceding elements of the sequences (u) and (v) are known.

2. Definition. A positive integer C is called a period of a sequence

$$(w) \quad w_0, w_1, \dots$$

if an integer $N \geq 0$ exists such that

$$w_{n+C} = w_n \quad (n = N, N+1, \dots).$$

The smallest positive period of the sequence is called the primitive period and will be denoted by C_w , the corresponding value of N by N_w . Obviously one has: If a sequence (w) has a period C then also its primitive period C_w exists and $C_w | C$. Conversely every multiple of the primitive period of a sequence is a period of the sequence.

The first relation (1) can be considered as a linear inhomogeneous difference equation for v . In this equation the term not depending on v is bounded; in fact its absolute value is $\leq (m-1) \sum_{h=0}^s |p_h|$. Since by condition III the characteristic polynomial of the sequence has no roots with absolute value ≥ 1 , every solution of this difference equation is bounded, i.e. the sequence (v) is bounded.

Since both sequences (u) and (v) appear to be bounded there is only a finite number of possible couples (u_n, v_n) , hence the set of these couples, i.e. the considered process is periodic. Then also the sequences (u) and (v) are periodic and after a little argument one finds that the primitive period $C = C_{uv}$ of the sequence of couples (u_n, v_n) is equal to the least common multiple $\{C_u, C_v\}$ of the periods C_u and C_v . We now prove the further resultant $C_v | C_u$, whence it follows that $C_{uv} = C_u$.

For $n \geq N_u$ from $(E^{C_u} - 1)u_n = 0$ it follows that

$$(E^{C_u} - 1)v_n = (E^{C_u} - 1)u_n = 0.$$

Further for $n \geq N_v$ one has

$$(E^{C_v} - 1)v_n = 0.$$

Consequently for $n \geq N = \max_{C_u, C_v}(N_u, N_v)$ one has $Gv_n = 0$, where G is the greatest common divisor of $(E^u - 1)V$ and $E^v - 1$. Now by condition III the resultant of V and $E^{C_v} - 1$ is equal to a constant $\neq 0$. Hence apart from a constant factor the polynomial G is equal to the resultant of $E^u - 1$ and $E^{C_v} - 1$, i.e. to $E^d - 1$, where d denotes the greatest common divisor (C_u, C_v) of C_u and C_v .

By the minimum property of the primitive period C_v one concludes $C_v = d$, hence $C_v | C_u$.

Remark. If on U the supplementary condition is imposed that it be relatively prime to every cyclotomic polynomial, then by a similar argument one finds $C_u | C_v$, hence $C_u = C_v = C_{uv}$. In general we shall not assume this supplementary condition holds, consequently in general we have only $C_v | C_u = C_{uv}$.

3. In order to deduce further properties we introduce the resultant M of the polynomials U and V . By condition II this resultant M is an integer $\neq 0$. By a property of resultants there exist polynomials X and Y with integer coefficients such that

$$(2) \quad M = XU + YV.$$

We now introduce the characteristic sequence (a) defined by

$$(3) \quad a_n = Xv_n + Yu_n \quad (n = 0, 1, \dots)$$

Then using (1) and (2) we have

$$(4) \quad Ua_n = XUv_n + YUu_n = (XU + YV)v_n = Mu_n$$

and similarly

$$(5) \quad Va_n = XVv_n + YVu_n = (XU + YV)u_n = Mu_n.$$

From (3) it follows that the sequence (a) is periodic and that

$$C_a | \{C_u, C_v\} = C_u. \text{ Further from (5) it follows that}$$

$$M(E^{C_a} - 1)u_n = V(E^{C_a} - 1)a_n = 0 \quad (n = N_a, N_a + 1, \dots),$$

hence $C_u | C_a$ and consequently $C_a = C_u$. Thus the characteristic sequence (a) has the same period as the process.

Writing (4) in the form

$$Va_n \equiv 0 \pmod{M}$$

we learn that the sequence (a) reduced mod M satisfies a linear recurring relation with characteristic polynomial V .

Now under certain restrictions the period of the mod M reduced sequence is

equal ¹⁾ to $c(V, M)$. Here $c(V, M)$ denotes the smallest positive integer with

$$X^{c(V, M)} \equiv 1 \pmod{V(X), M}.$$

These restrictions are:

A. $V(0)$ and M are relatively prime;

B. The resultant of $\frac{V(E)-V(X)}{E-X} a_0$ and $V(X)$ is relatively prime to M .

Calling the prime factors of $V(0)$ and of the resultant of $\frac{V(E)-V(X)}{E-X} a_0$ and $V(X)$ exceptional for the considered reduced recurring sequence we may state the result in the following form.

If M has no exceptional prime factors the period c of the reduced recurring sequence is equal to the exponent $c(V, M)$. If however M possesses singular prime factors then the period c satisfies

$$c(V, M') \mid c \mid c(V, M),$$

where M/M' contains only exceptional prime factors.

Since C denotes the period of the sequence (a) and c denotes the period of the mod M reduced sequence (a) one has obviously $c \mid C$, hence under the conditions A and B

$$(6) \quad c(V, M) \mid C.$$

In a similar manner from (4) one concludes

$$(7) \quad c(U, M) \mid C,$$

if also for the sequence with characteristic polynomial U the integer M possesses no exceptional prime factors.

In the following sections some cases are considered in which more can be said than the results (6) and (7), in particular cases are treated where either $C = c(V, M)$ or $C = c(U, M)$. It is obvious that the first relation holds if one has $C \mid c$, i.e. if for sufficiently large n from $a_n \equiv a_{n+c} \pmod{M}$ it follows that $a_n = a_{n+c}$.

4. It is not difficult to find cases with $C = c(U, M)$. We show that these occur for instance if $V = m$ and M has no exceptional primes for the recurring sequence (a) with characteristic polynomial U .

¹⁾ Confer H.J.A. Duparc, Divisibility properties of recurring sequences, p.44-50, Thesis, Amsterdam 1953.

A shorter proof of this property will be given in the author's paper Periodicity properties of recurring sequences II, to appear in the Proc. Kon. Ned. Ak. van Wetensch. 1954.

In fact, in this case one has from (1) for $n = 0, 1, \dots$

$$Uu_n = mv_n$$

and moreover

$$M = m, X = 0, Y = 1, a_n = u_n.$$

Thus we get

$$Ua_n = mv_n, \text{ i.e. } Ua_n \equiv 0 \pmod{m}.$$

Since by (1) one has $0 \leq u_n < m-1$ the sequence (u) is identical to the mod m reduced sequence (u) hence $C = c(U, M)$.

To this class of problems belong the recurring fractions, which are obtained when converting the fraction u_0/m into the number system to the base p . Then one has

$$(8) \quad pu_n = mv_n + u_{n+1} \quad (n = 0, 1, \dots).$$

It is not without interest to state here under what conditions m has no exceptional prime factors. The condition A requires $U(0) = p$ and m to be relatively prime, whereas condition B requires that

$$\frac{U(E)-U(X)}{E-X} u_0 = u_0 \text{ and } m$$

are relatively prime. Here one obtains the wellknown exceptional cases in the theory of recurring fractions.

The general kind of problems belonging to the case $m = V$, for which one has no longer necessarily $s = 1$, is merely the theory of linear recurring sequences reduced mod m . For many properties of their periods $c(U, M)$ the reader is referred to the above mentioned papers.

In this section a further property on C could be given without using the fact that the integers are ordered. Hence these results also hold if the elements of the sequence (u) do not belong to the residue set $0, 1, \dots, m-1$ mod m but belong to any arbitrary set of representants of the residue set mod m of the integers ²). In the following section however I could not deduce the results without using the order of the positive integers.

²) Consequently the above results hold also for sequences the elements of which belong to the more general ff-sets, introduced in the above mentioned papers.

5. We now consider cases where $C = c(V, M)$. As a first group of cases we take $V = mE - q$, where by property III we have $|q| \leq m-1$. Now (5) becomes

$$ma_{n+1} - qa_n = Mu_n \quad (n = 0, 1, \dots).$$

Putting $a_{n+c} - a_n = Mb_n$, where as before c denotes the period mod M of the sequence (a) we get for sufficiently large n (and in this section unless stated otherwise only such n will be considered)

$$mb_{n+1} - qb_n = u_{n+c} - u_n.$$

(9) First we prove $|b_{n+1}| \leq |b_n|$.

In fact otherwise we would have $|b_{n+1}| \geq |b_n| + 1$, whence we would derive the contradiction

$$m-1 \geq |u_{n+c} - u_n| = |mb_{n+1} - qb_n| \geq m|b_{n+1}| - |qb_n| \geq m + (m-|q|)|b_n| \geq m.$$

Now first consider the case $q > 0$. Then one has $b_n b_{n+1} \geq 0$, for from $b_n b_{n+1} < 0$ we would obtain the contradiction

$$m-1 \geq |u_{n+c} - u_n| = |mb_{n+1} - qb_n| = m|b_{n+1}| + q|b_n| \geq m+q.$$

From the relation $c|C$ derived in section 3 we get putting $C = rc$ and using the definition of b_n

$$(10) \quad 0 = a_{n+rc} - a_n = M(b_n + b_{n+c} + \dots + b_{n+(r-1)c}),$$

hence $b_n = 0$ and $a_{n+c} = a_n$, i.e. $c = C$.

Further consider the case $q < 0$. Then one has

$$(11) \quad b_n b_{n+1} \leq 0,$$

for from $b_n b_{n+1} > 0$ we would obtain the contradiction

$$m-1 \geq |u_{n+c} - u_n| = |mb_{n+1} - qb_n| = m|b_{n+1}| + |qb_n| \geq m+q.$$

If c is even we use the relation (10), which also holds in this case. Then we conclude as before $b_n = 0$, i.e. $c = C$.

If however c is odd we use the relation

$$0 = a_{n+2rc} - a_n = M(b_n + b_{n+c} + \dots + b_{n+(2r-1)c});$$

herefrom using (9) and (11) it follows that $b_n + b_{n+c} = 0$, hence $a_{n+2c} = a_n$, i.e. $C = 2c$.

The case $U = 1 - E^2$, $V = 3E + 2$ with $M = 5$, $c = 1$, $C = 2$ shows that $C = 2c$ can occur.

Also in another class of cases we can prove $c = C$, viz. in the cases $V = mE^t + 1$.

If $V = mE^t - 1$, then (5) becomes for $n = 0, 1, \dots$

$$(12) \quad (mE^t - 1)a_n = Mu_n,$$

hence

$$(m^r E^{rt} - 1)a_n = M \sum_{j=0}^{r-1} m^j E^{jt} u_n,$$

consequently

$$0 \leq m^r a_{n+rt} - a_n \leq M(m-1) \sum_{j=0}^{r-1} m^j = M(m^r - 1),$$

$$\frac{a_n}{m^r} \leq a_{n+rt} \leq M + \frac{a_n - M}{m^r},$$

hence if h is sufficiently large

$$1 \leq a_h \leq M,$$

provided $a_h \neq 0$.

From (12) for $a_n = 0$ one deduces $a_{n+t} = 0$ on account of $0 \leq u_n \leq m-1$. (Similarly from $a_n = M$ one deduces $a_{n+t} = M$).

If $a_n \neq 0$, $a_n \neq M$ from $a_{n+c} \equiv a_n \pmod{M}$ one concludes $a_{n+c} = a_n$ and $C = c$.

Here one has $X^t = 1/m \pmod{V(X)}$,

hence if e denotes the exponent of $m \pmod{M}$ one has $X^{et} \equiv 1 \pmod{V(X), M}$ and it is not difficult to prove the et is the smallest positive exponent with this property. ³⁾

Consequently $C = et = t c_m(M)$ and this holds also if some elements of the sequence (a) are equal to 0 or to M .

If $V = mE^t + 1$, then we get similarly

$$(m^r E^{rt} - (-)^r)a_n = M \sum_{j=0}^{r-1} (-)^{r-1-j} m^j E^{jt} u_n,$$

consequently

$$(-)^r a_n - M m^r \frac{m-1}{2} \leq m^r a_{n+rt} \leq (-)^r a_n + M m^{r+1} \frac{m-1}{2},$$

hence for sufficiently large r

$$-M \frac{m-1}{2} \leq a_{n+rt} \leq M \frac{m(m-1)}{2}.$$

Since the difference of the left and righthand side of this relation is equal to M for sufficiently large h from $a_{h+c} \equiv a_h \pmod{M}$ one deduces $a_{h+c} = a_h$ and $C = c$, provided one of the exceptional cases $a_h = \frac{-M}{m+1}$ or $\frac{mM}{m+1}$ holds. Since from $a_h = \frac{mM}{m+1}$ it follows $a_{h+t} = \frac{-M}{m+1}$ and conversely, we get for

³⁾ H.J.A. Duparc, loc. cit. theorem 45, p.40.

even $e' = c_{-m}(M)$ the result $C = c$ and for odd e' the result $C = 2c$.

6. Very simple applications of the results of the preceding section occur for linear U and V , say

$$U = E + p, V = mE - q. (0 < q < m).$$

These processes can be considered as an immediate generalisation of those on recurring fractions given in section 4, formula (8).

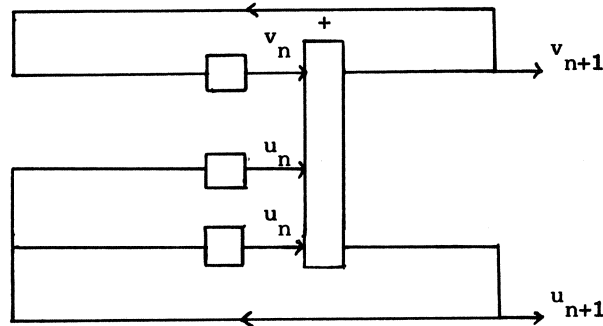
Here one has $M = |mp+q|$. The condition A here requires that q and M are relatively prime, i.e. that $(q, mp) = 1$; the condition B requires $(m a_0, M) = 1$. We assume these conditions satisfied, i.e. $(q, m) = (q, p) = (a_0, M) = 1$.

Here V is linear, hence the period $c(V, M)$ is equal to the exponent of $\frac{q}{m} \bmod M$ which will be denoted by $c_{q/m}(M)$. Then also the period of the process is equal to this number.

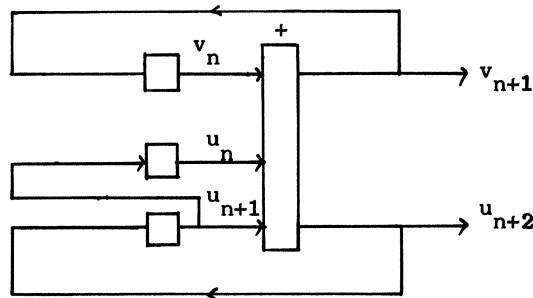
The process is given by the recurring equation

$$pu_n + qv_n = u_{n+1} + mv_{n+1} \quad (n = 0, 1, \dots),$$

which can easily be represented by a cyclic computing circuit, which works in the number system to the base m . We show below the case $p = 2, q = 1$.



Another interesting circuit is given by the computing network



The corresponding equation is

$$u_n + u_{n+1} + v_n = u_{n+2} + m v_{n+1} \quad (n = 0, 1, \dots).$$

Here

$$U = -E^2 + E + 1, \quad V = mE - 1, \quad M = m^2 + m - 1$$

and the condition A is obviously satisfied. The condition B requires

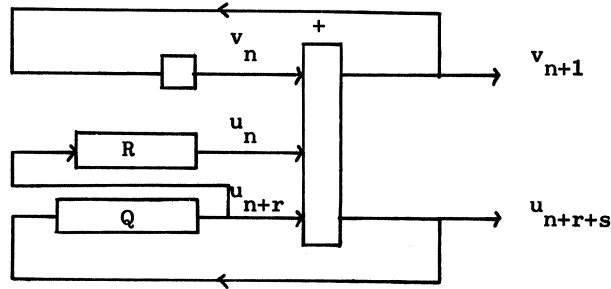
$$(a_0, m^2 + m - 1) = 1. \text{ Supposing B holds one has}$$

$$C = c_m (m^2 + m - 1).$$

As a last example we take the process

$$u_n + u_{n+r} + v_n = u_{n+s} + m v_{n+1} \quad (n = 0, 1, \dots; 0 \leq r < s),$$

realised by the computing network



Here Q and R denote delay lines with length $q = s - r > 0$ and r respectively.

We have

$$U = 1 + E^r - E^s, \quad V = mE - 1, \quad M = m^s + m^{s-r} - 1.$$

Since $V(0) = -1$ the condition A holds; the condition B holds if we assume ma_0 and M , i.e. a_0 and M relatively prime. Then one finds

$$C = c_{1/m}(M) = c_m (m^s + m^q - 1).$$

DE A.R.R.A.

G.A. Blaauw

Lezing in het colloquium Moderne Rekenmachines van het Mathematisch Centrum op 29 mei 1954 te Amsterdam.

1. Inleiding

De A.R.R.A. is een automatische cijfermachine werkend in het tweetallig stelsel. De bewerkingen worden in serie uitgevoerd en omvatten onder meer optelling, aftrekking, vermenigvuldiging en deling. De machine heeft een geheugencapaciteit van 1024 woorden van 30 cijfers. Een woord kan worden gebruikt als een getal van 29 binaire cijfers en een teken, of als een tweetal instructies van 15 binaire cijfers elk. Deze twee instructies worden achtereenvolgens en afzonderlijk uitgevoerd. Gegevens worden aan de machine toegevoerd met behulp van een handlezer voor een vijfsgatspapieren teletype band. Resultaten worden afgeleverd met behulp van een elektrische schrijfmachine. Een bedieningspaneel maakt het mogelijk op elk gewenst punt in de berekening in te grijpen, een berekening stap voor stap te controleren, en getallen in decimale vorm aan de machine toe te voeren.

De A.R.R.A. is ontworpen en geconstrueerd tussen 1 januari 1953 en 1 februari 1954. De machine maakt gebruik van seleengelijkrichters als schakelementen, magnetische kernen als geheugenelementen, terwijl elektronenbuizen worden gebruikt voor het leveren van de spanning en de energie die deze elementen vereisen. De magnetische kernen worden als snel geheugenelement in het arithmetisch gedeelte van de machine gebruikt. Hiernaast wordt een trommel met magnetisch oppervlak gebruikt welke de geheugencapaciteit van 1024 woorden levert. Relais worden gebruikt voor de selectie van de lees- en schrijfkoppen van de trommel en voor de verbinding van het electro-

nisch gedeelte van de machine met de schrijfmachine en de bedieningsknoppen.

Alle elektronische apparatuur is in plugeenheden gemonteerd, terwijl alleen de noodzakelijke verbindingsdraden vast op een rek zijn gesoldeerd. Op deze wijze kunnen, zo er een defect optreedt, de eenheden, waar de storing ligt, snel door andere worden vervangen. Ook kunnen door middel van een systematische controle zwakke elementen worden verwijderd voordat een storing optreedt.

2. Onderdelen

A. Gelijkrichters

Het grootste deel der logische schakelingen in de A.R.R.A. geschiedt door middel van seleengelijkrichtereenheden. De functie van ieder van deze eenheden kan worden opgevat als een "en - of" schakeling.

De werking hiervan kan worden ingezien aan de hand van een eenvoudige schakeling van gelijkrichters als in fig. 1.

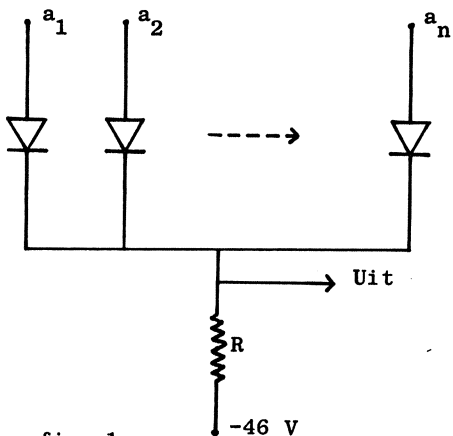
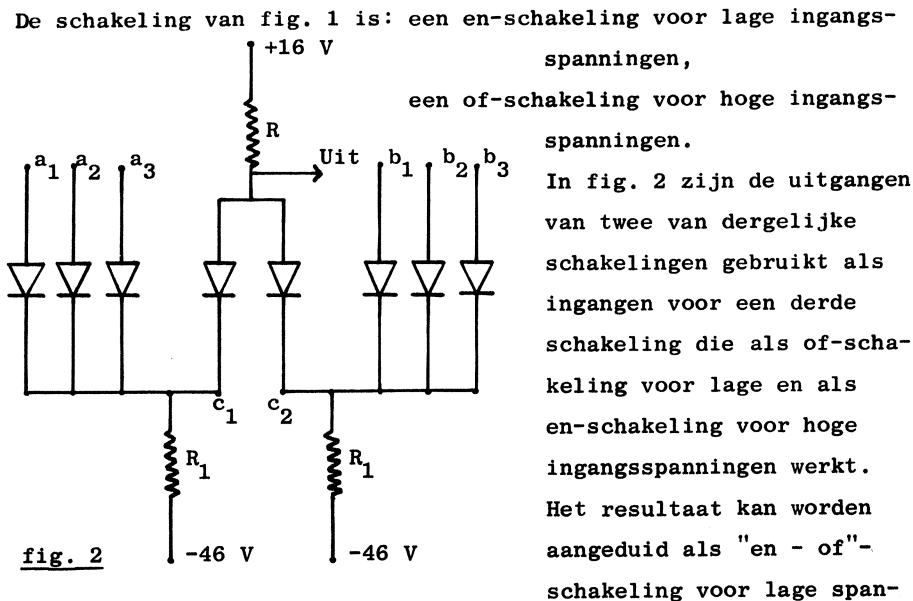


fig. 1

Aangenomen wordt dat de ingangsspanningen $a_1, \dots, a_n$ binair zijn, d.w.z. slechts twee bepaalde waarden kunnen aannemen, die voor dit geval op +3 V en -16 V zijn gesteld. Deze beide waarden worden resp. de "hoge" en de "lage" waarde genoemd.

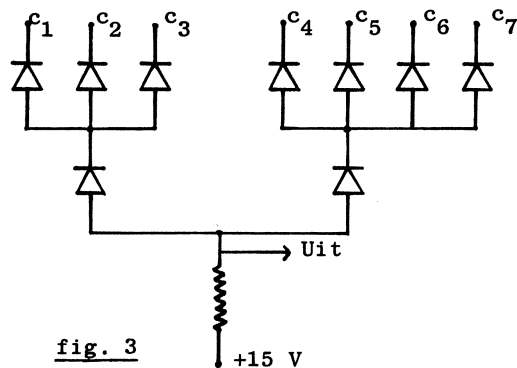
In de schakeling van fig. 1, zal de uitgangsspanning alleen laag zijn (-18 V) wanneer alle ingangsspanningen laag zijn. De

waarde van weerstand R wordt hier hoog verondersteld ten opzichte van de voorwaartse weerstand van de gelijkrichtcellen bij het werkpunt. De spanningval over de gebruikte cellen bedraagt ongeveer 2 V. De uitgangsspanning zal hoog zijn (+ 1 V) indien een van de ingangsspanningen hoog is (+3 V).



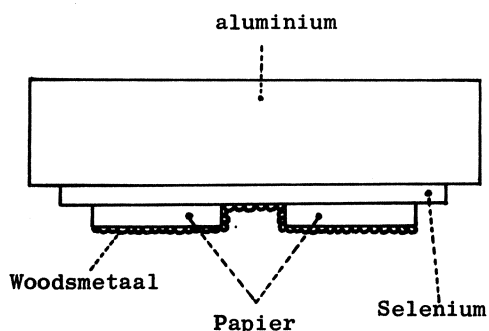
ningen. Immers de uitgangsspanning zal laag zijn als: a_1 en a_2 en a_3 , of b_1 en b_2 en b_3 , of beide groepen laag zijn.

De gelijkrichters a_1 , a_2 en a_3 behoren tot eenzelfde "niveau" evenals b_1 , b_2 en b_3 . Aan de schakeling van fig. 2 kunnen meer niveaus worden toegevoegd. De gelijkrichters met c_1 , c_2 , ... als ingangen worden de combinatie-gelijkrichters genoemd. In eenheden met een groter aantal niveaus is de opbouw van de combinatiegelijkrichters vaak iets uitgebreider. Doel hiervan is het versnellen van de werking, door het verminderen van de totale capaciteit die de gelijkrichters vertegenwoordigen. In dit geval worden de niveaus in groepen van 3 of 4 gecombineerd, terwijl de groepen op hun beurt weer gecombineerd worden (zie fig. 3). De gelijkrichters, welke thans ge-



bruikt worden voor de samenstelling van deze eenheden zijn opgebouwd uit een aluminium schijf van 6,3 mm. diameter, aan een zijde voorzien van een laagje selenium. Deze seleniumlaag is bedekt

met een laag papier met een kleine opening in het midden, terwijl op de buitenkant van het papier een laagje Woods-metaal is aangebracht voor het electrisch contact met het selenium. Het doel van de papierlaag is het verkleinen van de actieve oppervlakte van het selenium, teneinde de capaciteit van de cel te verminderen (fig. 3a). De karakteristieke gegevens op de werkpunten van de gelijkrichters zijn:



0,7 mA voorwaartse stroom bij
2 V voorwaartse spanning
1 A tegenwaartse stroom bij 15 V
tegenwaartse spanning
3 k.ohm en 15 M.ohm voorwaartse
resp. tegenwaartse weerstand.

(waarden van 10 k.ohm resp. 2 M.ohm zijn nog toelaatbaar).

Tengevolge van de betrekkelijk hoge capaciteit (70 pF per gelijkrichter) is de snelheid der seleengelijkrichtereenheden begrensd; de hoogste frequentie die bij een blokspanning toegepast kan worden is van de orde van grootte van 100 kHz. Bij spanningssprongen is de vertraging voor een eenheid ongeveer 6 μ S.

Een andere belangrijke eigenschap van dit type schakelingen is, dat zij niet geschikt zijn voor het leveren van vermogen van enige betekenis aan een belasting. Derhalve wordt iedere gelijkrichter-schakeling gevolgd door een versterker met kathodevolgeruitgang om de vorm van het energieniveau van het signaal te herstellen.

B. Buiseenheden

Vele van de buiseenheden in de A.R.R.A. hebben geen logische functie, maar dienen om vorm en energie-niveaus van de uitgangssignalen der seleeneenheden te herstellen. Zij bestaan doorgaans uit direct gekoppelde versterkers met kathodevolger-uitgangen. De signalen in de machine zijn veelal in beide polariteiten beschikbaar en worden aangeduid als x en x'. Hierbij is dan als de numerieke waarde van het signaal 0 is x laag en x' hoog. Als de numerieke waarde 1 is, is x hoog en x' laag.

Elke eenheid van de machine heeft een type aanduiding en een serienummer en zijn levensloop kan worden nagegaan aan de hand van een logboek en kaartstelsel.

Op de kaarten worden de resultaten van de metingen betreffende de vastgestelde veiligheids marges gevonden. Deze metingen worden op speciaal voor dit doel gebouwde testers verricht.

C. Statische magnetische registers

De machine beschikt over drie snelle registers, welke gebruik maken van magnetische kernen. De registers genaamd A, M en S hebben 59, 59 en 57 kernen en kunnen resp. 30, 30 en 29 cijfers bevatten.

D. Magnetische trommel

De 1024 woorden welke het geheugen kan bevatten, zijn geplaatst als magnetische impulsen op het oppervlak van een snel draaiende trommel. De trommel heeft de vorm van een riemschijf met brede flens en draait om een vaststaande as. De trommel wordt aangedreven met behulp van een snaar en heeft een snelheid van 3000 omwentelingen per minuut. De breedte van de flens is 160 mm, de diameter bedraagt 300 mm. Momenteel bevinden zich 40 kanalen op de trommel. Signalen worden op de trommel geschreven en er van gelezen met behulp van koppen ontwikkeld voor dit doel door de P.T.T..32 van de 40 beschikbare koppen zijn aangesloten aan de machine. Op elk kanaal kunnen 32 woorden geschreven worden. Hierbij zijn 8 woorden aaneensluitend rond de trommel omtrek geplaatst, terwijl de magnetische impulsen behorende bij vier van zulke groepen van 8 geïnterspatieerd zijn. Elke vierde puls behoort dus bij eenzelfde woord. De trommel is van aluminium gemaakt en heeft een rood-ijzer oxide coating. Deze coating is door de N.V. Philips Gloeilampenfabrieken te Eindhoven ontwikkeld. Op de trommel is tevens een stalen klokpuls-ring gemonteerd waarop 512 verdelingen zijn gegraveerd. Hiervoor worden de impulsen afgeleid die gebruikt worden voor de tijdsignalen van de machine.

Blokschema

In afbeelding 4 is een blokschema voor de A.R.R.A. weergegeven. Het onderlinge verband tussen de vier gedeelten van de machine:

a. het geheugen, b. de besturing, c. het arithmetisch gedeelte en d. in- en uitvoer en bediening wordt hierin geïllustreerd.

a. Het geheugen. De 32 kanalen van de magnetische trommel kunnen onafhankelijk worden geselecteerd door de opdracht relais 0 en de getal relais G. Elke groep bestaat uit 9 relais voorzien van 4 wisselcontacten voor de selectie van trommel signalen en 2 wisselcontacten voor de controle van de stand van het relais. De trommelsignalen worden versterkt door de versterkers OU en GU. Tevens worden in deze schakelingen de gewenste impulsen geselecteerd. De controle signalen worden gebruikt in het "wacht" circuit. Hier wordt de stand van de relais vergeleken met de stand van de drijfhuizen voor de relais. Zolang de relais nog in beweging zijn wordt een wacht ingevoerd. De opdracht selectie wordt alleen gebruikt voor lezen, de getal selectie zowel voor lees-signalen als schrijf-signalen. De laatsten zijn afkomstig van de eenheid STC.

b. De besturing. De drie voornaamste onderdelen van de besturing zijn het opdrachtregister (OR), de opdrachtteller (OT) en de controlering (CR). Het opdrachtregister bestaat uit vijftien triggerpairs die tezamen een opdracht kunnen bewaren. De opdracht wordt vanuit OU in het register geplaatst. Vijf cijfers van de opdracht dienen voor de aanduiding van de door het arithmetisch gedeelte te verrichten bewerking, dit is het "functie gedeelte" van de opdracht. De overige tien cijfers dienen voor de aanduiding van de plaats in het geheugen waar het getal staat dat bij de bewerking betrokken is, dit is het "adres gedeelte". De opdracht blijft tijdens de bewerking onveranderd in het register staan. De opdracht teller bestaat uit elf trigger pairs, die tezamen het adres bewaren van de opdracht die zal worden uitgevoerd. Tien van deze trigger pairs controleren de opdracht relais en de plaats en impulsselectie van OU, een elfde eenheid is aanwezig om aan te duiden welke helft van het geselecteerde woord wordt gebruikt.

Het adres geplaatst in de opdrachtteller wordt na elke gelezen opdracht opgehoogd. Bovendien kan voor een sprong-opdracht het adresgedeelte van het opdrachtregister in de opdrachtteller worden geplaatst.

De opeenvolging van de stappen: opdracht lezen, opdrachtteller ophogen, nieuwe opdracht selecteren en getal selecteren, getal lezen of schrijven en bewerken wordt bepaald door de Controle Ring.

Behalve deze reeks verzorgt de controle ring tevens de opeenvolgende stappen nodig voor de door de handbediening vereiste bewerkingen.

De controlering is een gelijkrichter ring bestaande uit vier trigger pairs welke de mogelijke 16 standen in enige specifieke volgorden kunnen doorlopen. De bedoelde volgorden zijn verschillend afhankelijk van de uit te voeren bewerkingen en worden bepaald door de handbediening en het opdrachtregister.

c. Het arithmetisch gedeelte. De drie registers A, M en S tezamen met hun ingangen AI - SI, uitgangen AU - SU, drijfcircuits AD - SD en tekenregisters TA - TS vormen het voornaamste gedeelte van het arithmetisch gedeelte. In elk van de genoemde circuits zijn inbegrepen de controle voorwaarden nodig voor het juist functioneren van deze registers gedurende de door het opdracht register aangegeven bewerkingen. Een lijst van opdrachten is aangegeven op blz. 7. Register M is uitsluitend gebruikt in de vermenigvuldig, deel en "maal tien" bewerkingen. Registers A en S kunnen afzonderlijk als accumulatoren worden gebruikt. Bij deze registers behoren de overdrachtregisters OA en OS, welke tezamen met VO tijdens de optelling worden gebruikt. Na afloop van de bewerking kunnen deze registers de eindoverdracht bevatten. Het getal x kan dus op twee wijzen in A geplaatst zijn, namelijk met x in het magnetisch register en 0 in OA of met x-1 in het magnetisch register en 1 in OA. Na elke schrijf bewerking bevindt zich steeds 0 in OA. Dezelfde opmerkingen kunnen voor OS gemaakt worden. OA en OS worden tevens tijdens de vermenigvuldiging en deling gebruikt. De aanwezigheid van een overdracht wordt bepaald op grond van de signalen afkomstig van de registers (US), het geheugen (LTC) en de overdracht van het vorige cijfer (VO). Deze drie signalen worden tevens gebruikt om het eenheden cijfer van de som te bepalen in OV.

d. In- en uitvoer. Voor de uitvoer is beschikbaar een elektrische schrijfmachine. De toetsen van deze schrijfmachine worden door electromagneten aangetrokken. De selectie van de aan te trekken toetsen vindt plaats door een relaisboom verbonden met de trigger pairs U1-U8. Gegevens kunnen worden toegevoerd aan de machine met behulp van een bandlezer en met de handbedieningsknoppen en schakelaars. Deze signalen worden gecombineerd in de schakeling LS. Aan de schakeling LS wordt tevens toegevoerd een controlesignaal afkomstig van de schrijfmachine relais.

De bedieningsknoppen stellen de operator in staat een programma op een gekozen adres te beginnen of te stoppen of dit op het volgende adres te doen, tevens om een enkele opdracht uit te voeren of te lezen of een opdracht vanaf het bedieningsbord te introduceren en uit te voeren.

Opdrachten A.R.R.A.

0/n	vervang (A) door $(A) + (n)$
1/n	vervang (A) door $(A) - (n)$
2/n	vervang (A) door (n)
3/n	vervang (A) door $-(n)$
4/n	vervang (n) door (A)
5/n	vervang (n) door $-(A)$
6/n	conditionele besturingsverplaatsing naar n a
7/n	besturingsverplaatsing naar n a
8/n	vervang (S) door $(S) + (n)$
9/n	vervang (S) door $(S) - (n)$
10/n	vervang (S) door (n)
11/n	vervang (S) door $-(n)$
12/n	vervang (n) door (S)
13/n	vervang (n) door $-(S)$
14/n	conditionele besturingsverplaatsing naar n b
15/n	besturingsverplaatsing naar n b
16/n	vervang $[AS]$ door $[n] \cdot [S] + [A]$
17/n	vervang $[AS]$ door $-[n] \cdot [S] + [A]$

- 18/n vervang $[AS]$ door $[n] \cdot [S]$
 19/n vervang $[AS]$ door $-[n] \cdot [S]$
 20/n deel $[AS]$ door $[n]$, plaats quotiënt in S, rest
 in A
 21/n deel $[AS]$ door $-[n]$, plaats quotiënt in S, rest
 in A
 22/n "schuif $A \rightarrow S$ ", d.w.z. vervang $[AS]$ door $[A] \cdot 2^{29-n}$
 23/n "schuif $S \rightarrow A$ ", d.w.z. vervang $[SA]$ door $[S] \cdot 2^{30-n}$
 24/n communicatie opdracht.

Communicatie opdrachten

- 24/0 en 24/1 "skip-opdracht"
 24/2 "+ conditionele stop"
 24/3 "- conditionele stop"
 24/4 "lees band"
 24/5 "vervang $[A]$ door $-[H]$ "
 24/6 "vervang (A) door (OT)"
 24/7 "vervang (A) door (GS)"
 24/8 "type-opdracht"
 24/12 "vervang $[A]$ door $[U]$ "
 24/16 "vervang $[AS]$ door $10 \times [S] + [A]$ "
 24/17 "vervang $[AS]$ door $10 \times [S]$ "
 24/n "als $512 \leq n \leq 767$, vervang $[A]$ door
 $[A] + n - 512$ "
 als $768 \leq n \leq 1023$, vervang $[A]$ door
 $[A] + n - 1023$ "

