Beveiliging van een Read Only Storage

Huits, M.H.H.

Award date:
1968

TECHNISCHE HOGESCHOOL EINDHOVEN

AFDELING DER ELECTROTECHNIEK

BEVEILIGING VAN EEN READ ONLY STORAGE

Afstudeerverslag van

M.H.H.Huits

Groep ECB, november 1968

Afstudeeropdracht uitgevoerd in de groep Telecommunicatie B (Prof. Ir. A. Heetman) Onder leiding van Ir.C.P.J.Schnabel (T.H.E.) en Ir. W.G.J. Kreuwels (N.V. Philips Natuurkundig Laboratorium).
januari - november 1968.

Voorwoord

De schrijver van dit verslag kreeg de opdracht een studie te maken van de mogelijkheden die de literatuur kent om een stuk logica tegen het maken van fouten te beveiligen. Deze logica noemen we "hard-core".

In een computer kan deze logica gebruikt worden om andere delen van de machine te testen.

Een voor hard-core bij uitstek geschikt deel van de computer is de Read Only Storage. Daarom zijn de gevonden methoden, voor zover mogelijk, toegepast op de beveiliging van een Read Only Storage.

De schrijver is dank verschuldigd aan
Prof. Ir. A. Heetman, Ir. C.P.J.Schnabel, Ir. P. Klijzing en in het bijzonder Ir. W.G.J. Kreuwels voor de waardevolle opmerkingen die hebben bijgedragen tot de tot standkoming van dit verslag.

Inhoudsopgave pag.

1. Inleiding.

Naast de verwerkingssnelheid van data in electronische rekenmachines, neemt vooral ook de betrouwbaarheid waarmee bepaalde gegevens moeten worden verwerkt, een steeds belangrijker plaats in. Daar er steeds hogere eisen gesteld worden aan de verwerkingsmogelijkheid van zeer ingewikkelde processen, groeit ook de complexheid en daarmee het aantal bouwstenen (nands, flip-flops etc.) van de rekenmachine. De kans op het defect raken van één enkele bouwsteen neemt, ondanks de hoge kwaliteit van deze bouwstenen, steeds toe. Het is dan ook noodzakelijk aandacht te besteden aan maatregelen die genomen kunnen worden om de foutloze verwerking van gegevens te waarborgen.

Een rekenmachine bestaat uit vele units, zoals geheugens en registers. In grote lijnen kunnen we de organisatie van een machine als volgt voorstellen. (fig.1).
De communicatie met de buitenwereld geschiedt met input-output-devices, zoals kaartlezer, ponser en printer. Deze IO-devices zijn aangesloten op control-units. In de control-units vindt een herkenning plaats van de betekenis van de aangeboden informatie en een omzetting van deze informatie naar de machinecode. Bovendien worden wachttijden, ter synchronisatie van de I-O-devices met de rest van de machine, hierdoor geregeld. De communicatie verloopt verder via Channels.
Het centrale geheugen M dient om informatie gedurende kortere of langere tijd te bewaren.
In de central processing unit (CPU) wordt alle informatie verwerkt. Bepaalde standaardbewerkingen binnen de machine worden rechtstreeks vanuit de CPU geleid. Ze worden dan in de vorm van microprogramma's in een permanent geheugen opgenomen. Deze programma's behoeven, nadat ze in het geheugen zijn opgenomen, vrijwel nooit meer gewijzigd te worden. Een geheugen dat zich uitstekend leent voor de registratie van een dergelijk programma is de Read Only Storage (ROS).

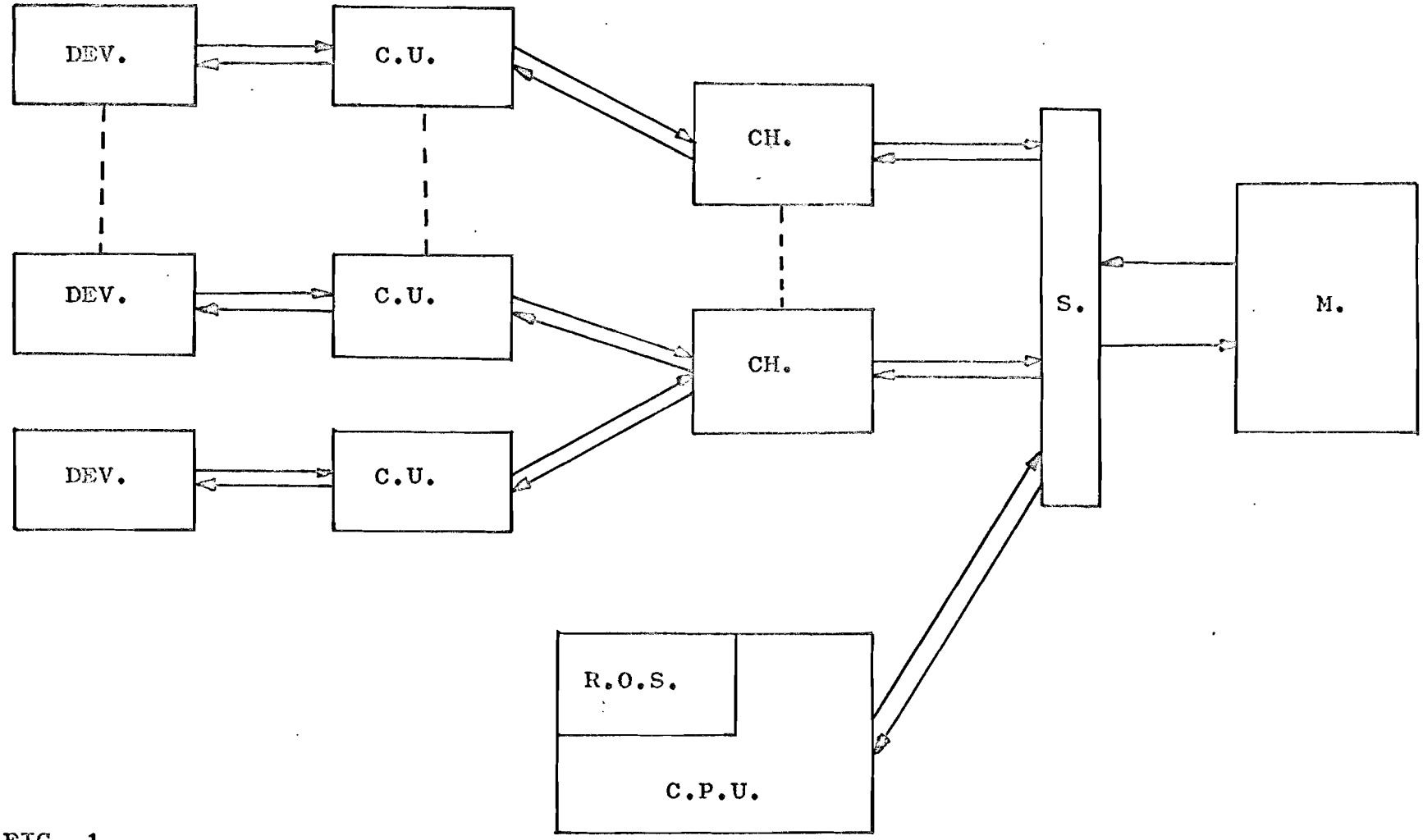


FIG. 1

Als in een van de genoemde eenheden een fout ontstaat, dan kan er bij bepaalde bewerkingen een foutieve output gegeven worden. Dit is vanzelfsprekend niet gewenst. Alle eenheden voldoende corrigerend maken, zou hier een oplossing bieden. Deze zal echter ook maar uitkomst bieden, zolang het aantal fouten de correctiemogelijkheden niet te boven gaat. Deze oplossing is bovendien erg duur.
Men heeft naar andere wegen gezocht om tot een beveiliging te komen. Een van de gevonden mogelijkheden is het op regelmatige tijden controleren of nog alles correct functioneert. Dit kan gebeuren met behulp van testprogramma's, die aan de afzonderlijke logische eenheden inputcombinaties aanbieden en dan de verkregen outputs vergelijken met de, van tevoren bekende, corresponderende outputs. Bij dit vergelijk kan geconstateerd worden wanneer er ergens een fout gemaakt is. Door geschikte keuze van de testprogramma's is het bovendien mogelijk localisatie van een bepaalde fout te verkrijgen. We noemen dit machine-diagnostiek.
Omdat een kern van de diagnostiekprogramma's niet aan wijzigingen onderhevig is, kan deze met voordeel als microprogramma in de ROS worden ondergebracht. De operateur kan hiermee de machine testen, en men kan tevens tot een snellere localisatie van eventuele fouten komen.
De testprogramma's worden in het centrale geheugen ingelezen. Wil het testprogramma waardevol zijn, dan zullen er in ieder geval geen fouten mogen ontstaan bij het aanbieden van informatie aan de logische eenheden. Het is daarom noodzakelijk de microprogramma's en dus de ROS te beveiligen, zodat we zekerheid hebben omtrent de juistheid van de aangeboden informatie. Dit afstudeerwerk heeft dan ook als doel:
mogelijkheden onderzoeken, die een effectieve beveiliging van de ROS waarborgen.

2. De "Read Only Storage".

Onderstaand volgt een beschrijving van een ROS.
Daar een concrete gedetailleerde beschrijving van een ROS zeer complex is en bovendien weinig mogelijkheden biedt voor een algemene aanpak van het gestelde beveiligingsprobleem, beperken we ons tot de behandeling van een schematische versie.
Om toch enige indruk te geven omtrent de grootte van een concreet voorbeeld, zijn aan het eind van dit hoofdstuk enkele cijfers hierover vermeld.
De ROS (fig.2) bestaat uit, een geheugen, een selectieregister, een selectie-decodatie, een conditie-register en een tijden-teller.
Het geheugen is het centrale deel van de ROS. Hierin wordt alle informatie bewaard. Het is niet-destructief en woordgeorganiseerd.
De keuze van een bepaald woord wordt bepaald door de inhoud van het selectieregister.
De selectie-decodatie verzorgt de bekrachtiging van de, met de inhoud van het selectieregister overeenkomende, woordlijn.
Het uitgelezen woord verschijnt in de uitleesbuffer. Deze bestaat uit evenveel flip-flops als het woord bits telt. Een gedeelte van het uitgelezen woord bevat informatie over het adres van het volgende woord en wordt daarom in het selectieregister geplaatst.
Soms is echter de keuze van het nieuwe adres afhankelijk van situaties buiten de ROS. Dan wordt het conditieregister met externe informatie gevuld. Bij selectie van het nieuwe adres neemt deze informatie de plaats in van enkele bits in het selectieregister.
Een groot gedeelte van de informatie in de uitleesbuffer dient voor de bekrachtiging van commandolijnen. Commandolijnen die nooit gelijktijdig bekrachtigd worden zijn in een bepaalde groep, voortaan "veld" genaamd, ondergebracht.
Zo is de inhoud van het uitleesregister opgebouwd uit een aantal velden. Ieder veld omvat een aantal flip-flops, dat groter of gelijk is aan $^{2}\log$ van het aantal bijbehorende commandolijnen.
De commando-decodatie zorgt voor de bekrachtiging van een commandolijn, behorende bij de inhoud van een veld.

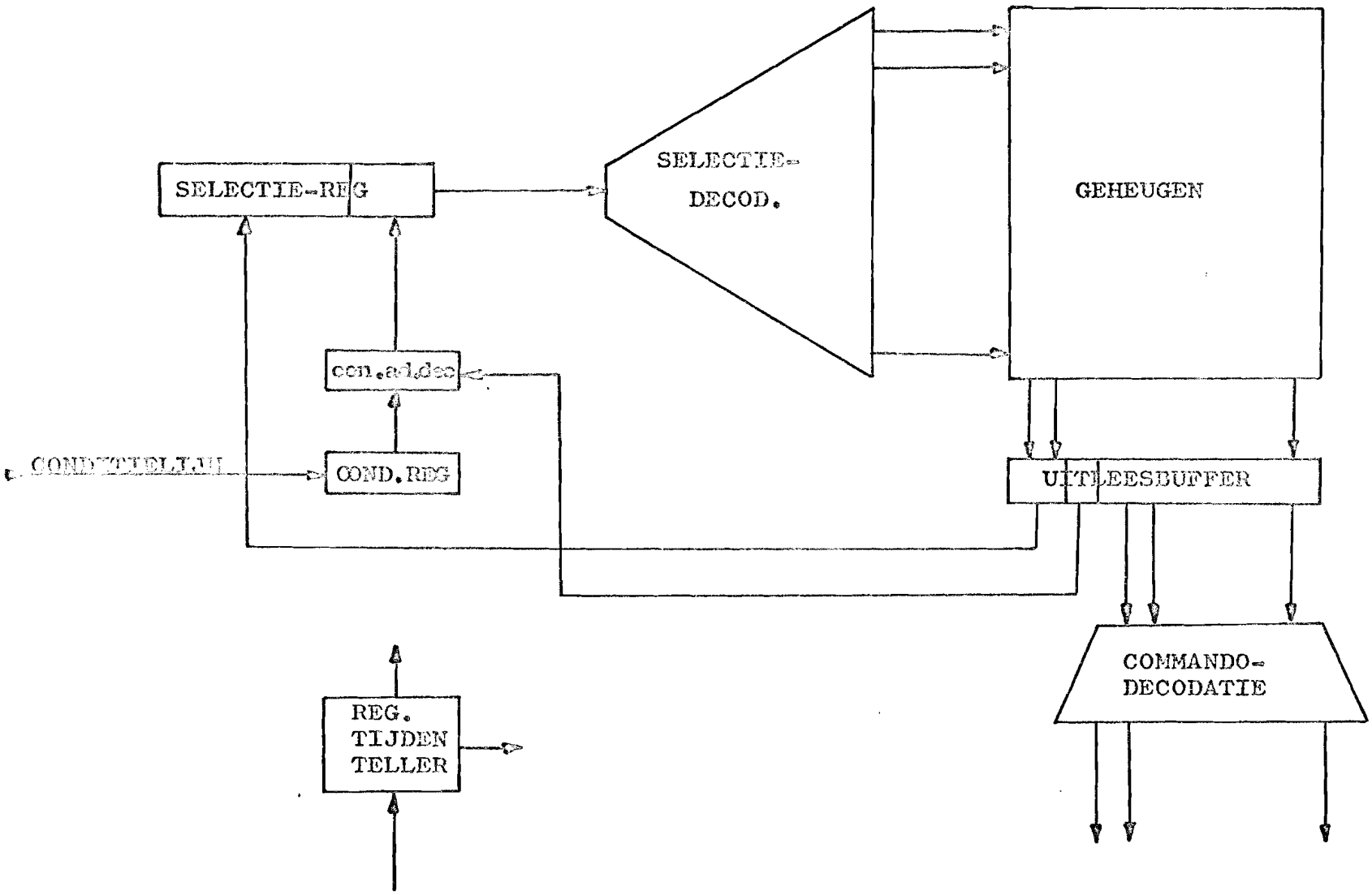

SELECTIE-REG
SELECTIE-
DECOD.
GEHEUGEN
con.adres
COND.REG
UITLEESBUFFER
COMMANDO-
DECODATIE
REG.
TIJDEN
TELLER
CENT.CLOCK

FIG. 2

Om zowel de commando's met het externe gedeelte te synchroniseren, als de interne informatie-stroom te controleren, is een register-tijdenteller aangebracht. Deze wordt gevoed vanuit de centrale klok, en geeft synchronisatie-signalen aan de diverse registers binnen de ROS.

Om een indruk te geven van de grootte-orde van de ROS:
de cyclustijd, dit is de tijd tussen het beschikbaar-zijn van twee opeenvolgende adressen, wordt geacht te liggen tussen de 100 en 500 nsec.
Het aantal woorden ligt tussen 1k en 30k.
Het aantal bits in een woord ligt tussen 100 en 200.

3. Het beveiligingsprobleem.

Een ROS is een samengesteld logisch systeem. Om te komen tot een beveiliging van een ROS, zullen we de beveiliging van logische circuits in het algemeen aan een onderzoek onderwerpen.
Wij zullen zo trachten richtlijnen voor het ontwerp bij de beveiliging van logica te geven. Het is hierbij in eerste instantie belangrijk na te gaan tegen welke categorie fouten we het systeem wensen te beveiligen. Bovendien zullen we beoordelingsnormen op moeten stellen, die een vergelijk tussen de verschillende oplossingen mogelijk maken. In een logisch systeem wordt informatie bestaande uit bits verwerkt. Een bit is de kleinst voorkomende informatie-eenheid. Deze kan de waarden 0 en 1 aannemen. Door uitwendige oorzaken, dan wel door onvolkomenheden in het systeem, kunnen deze bits van waarde veranderen. Hierdoor kan informatie verloren gaan of foutief geïnterpreteerd worden. Oorzaken die de ongewenste verandering van een bit tot gevolg kunnen hebben, noemen we fouten. De oorzaken van deze fouten kunnen velerlei zijn, terwijl ze ook verschillende eigenschappen kunnen hebben. We zullen ze daarom nader bestuderen en in een aantal categoriën indelen. Daarna zullen we beoordelingsnormen voor beveiligingen opstellen. (6).

3.1. Onderscheid naar de aard van fouten.

We noemen het te onderzoeken logische systeem "machine M".
Een machine M is een apparaat met ingangen en uitgangen en eventueel interne toestanden, die allen verschillende waarden kunnen aannemen. Deze waarden zijn alléén gedefinieerd voor een reeks tijdstippen t_o, ..., t_v-1, t_v, t_{v+1}, ... Met andere woorden machine M is synchroon.
We veronderstellen te beschikken over een beschrijving van machine M wanneer er geen fouten worden gedacht aanwezig te zijn. Indien de realisering onder alle omstandigheden en voor alle variabelen voldoet aan deze beschrijving, dan noemen we deze machine MO. In alle andere gevallen hebben we te maken met MI.
Fouten kunnen we nu als volgt indelen:

3.1.1. Design failures, of fouten die het gevolg zijn van een verkeerd ontwerp.

3.1.1.1. Logische fouten. De realisatie van de machine komt niet overeen met de specificatie van het ontwerp.

3.1.1.2. Hazards. Dit is een veel voorkomend fouttype, dat vaak moeilijk te ontdekken is en alleen door een zorgvuldig ontwerp vermeden kan worden.
We kunnen hazards onderscheiden in twee soorten. De statische en de dynamische hazard.
Een statische hazard (fig.3) ontstaat, wanneer twee elkaar onmiddellijk in de tijd opvolgende input-toestanden, die elk een en het-zelfde uitgangsniveau tot gevolg hebben, bij hun opvolging gedurende korte tijd een wisseling van het uitgangsniveau kunnen veroorzaken.
Een dynamische hazard ontstaat wanneer twee elkaar onmiddellijk in de tijd opvolgende input-toestanden die elk een verschillende uitgangsniveau tot gevolg hebben bij hun opvolging gedurende korte tijd meerdere niveau-wisselingen van de uitgang tot gevolg kunnen hebben.(fig.4).

3.1.1.3. Kritische races. In sequentiele netwerken kan een volgende stabiele toestand op niet-eenduidige wijze bereikt worden. Het is hierbij mogelijk dat uitgaande van gelijke begincondities, via bepaalde tussentoestanden, verschillende eind-toestanden bereikt kunnen worden. We hebben dan te maken met een critische race, die door zijn niet-eenduidigheid oorzaak van fouten kan zijn.

3.1.1.4. Dynamische fouten. Indien de verwerkingssnelheid van een machine te hoog is, kan dit tot fouten aanleiding geven. Deze categorie noemen we dynamische fouten.

3.1.1.5. a-logische fouten. Als gevolg van temperatuurschommelingen, vochtigheidsgehalte, netspanningsvariaties etc. kunnen ook fouten geintroduceerd worden. Ze zijn vaak moeilijk te detecteren. We noemen ze a-logische fouten.

We laten verder bovenstaande fouttypen buiten beschouwing, daar ze buiten het terrein van het onderzoek vallen. Doorgaans dienen zij te worden opgevangen door correcte ontwerpregels.

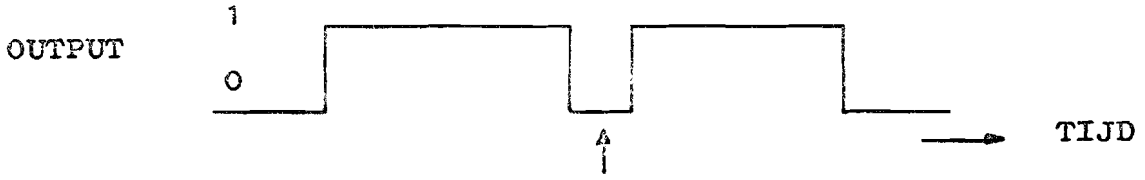
3.1.2. Fouten die niet het gevolg zijn van een verkeerd ontwerp. Indien aan het ontwerp optimale zorg besteed is, kunnen er toch nog fouten ontstaan ten gevolge van uitwendige storingen of verouderingsprocessen. Het is dan onze taak deze fouten te detecteren of eventueel te corrigeren.
Een essentieel verschil tussen de verschillende storingsvormen is, of zij van tijdelijke en dus voorbijgaande aard zijn, dan wel van ruimtelijke en dus permanente aard zijn.
Tijdelijke storingen kunnen vaak, na detectie, door het stellen van een herhalingsvraag ondervangen worden. Ruimtelijke storingen kunnen alleen ruimtelijk geëlimineerd worden. Een ruimtelijke beveiliging biedt voor zowel tijdelijke als ruimtelijke storingen een oplossing. Mijten we de ontstane fouten aan mutaties in de samenstellende componenten van machine M, dan kunnen we de volgende gevallen onderscheiden.

3.1.2.1. De machine blijft MO voor alle rangnummers v uit de reeks $t_o \ldots t_v$. Dit duidt op redundantie van de betrokken component of een deel daarvan.

3.1.2.2. Vanaf zekere v en voor alle daarop volgende waarden wordt de machine MI. De mutatie in kwestie noemen we een persistente fout. (fig.5a).

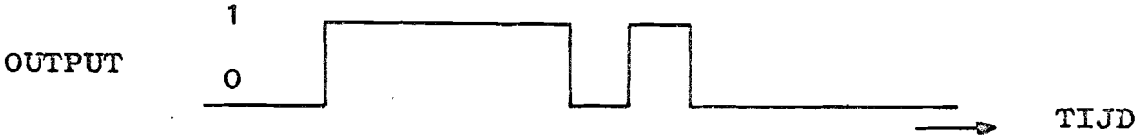
3.1.2.3. Vanaf zekere v vinden we willekeurig afwisselend MI en MO. Deze mutatie noemen we een intermitterende fout (fig.5b).

3.1.2.4. Vanaf zekere v vinden we voor een aaneengesloten aantal waarden voor v M1, doch voor de daarop volgende waarden van v weer MO. We hebben dan te maken met een transiente fout.(fig.5c).



WISSELING UITGANGS-NIVEAU

FIG. 3



MEERDERE UITGANGS-NIVEAU WISSELINGEN

FIG. 4

FIG. 5a

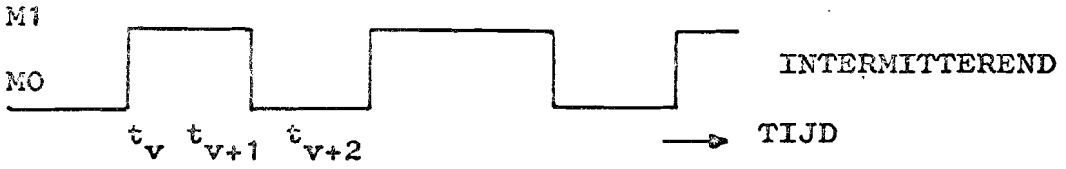


FIG. 5b

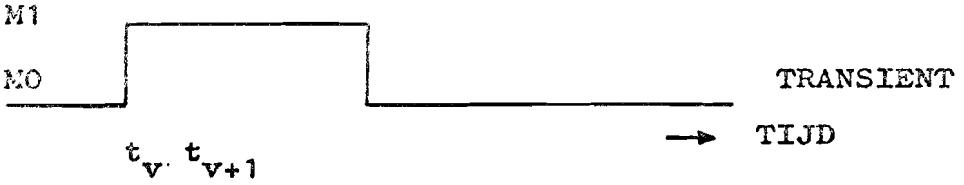


FIG. 5c

Bij het bestuderen van fouten kunnen we ook kijken naar de veranderingen, die er in de betrokken bits optreden. Is de kans, dat een bit van de waarde 0 naar de waarde 1 verandert, even groot als de kans, dat een bit van 1 naar 0 verandert, dan hebben we te maken met een symmetrische fout. Zijn deze kansen ongelijk, dan hebben we een a-symmetrische fout.
In samenhang met meerdere bits onderscheiden we onafhankelijke fouten, als de fouten rondom verdeeld zijn. Indien dit niet het geval is hebben we afhankelijke fouten, bijvoorbeeld "bursts". Deze laatsten spelen een voorname rol in data-communicatie en zijn van geen belang in het beschreven ROS-systeem. Zij ontstaan bij communicatiesystemen onder invloed van atmosferische storingen, die langer dan een bitlengte duren. Er zijn speciale codes ter beveiliging hiertegen ontwikkeld.

3.2. Beoordelingsnormen voor beveiligingen.

We stelden reeds dat we ons niet zullen beperken tot het bestuderen van de mogelijkheden die er zijn voor de beveiliging van een specifiek voorbeeld, maar ons bepalen tot het vinden van richtlijnen voor de beveiliging van een algemeen logisch systeem. Als we overwegen om over te gaan tot de beveiliging van logische systemen tegen fouten, zullen we op verantwoorde wijze een keuze moeten kunnen maken uit diverse beveiligingsmogelijkheden die ons ten dienste staan. Daartoe is het noodzakelijk eerst de beveiligingsmethoden aan een kritisch onderzoek te onderwerpen. Om dit of efficiënte wijze te kunnen doen geschieden, moeten we een aantal objectieve toetsingscriteria kunnen hanteren.
Onderstaand zijn parameters gegeven waaraan een systeembeveiliging getoetst kan worden.

3.2.1. Mogelijke toepasbaarheid met betreffing tot de aard van het systeem.

Sommige methoden lenen zich alleen voor de beveiliging van informatie-transporterende systemen, terwijl andere meer voor informatie-verwerking in aanmerking komen.

3.2.2. Het resultaat van de beveiliging.

3.2.2.1. Het effect van de beveiliging kan zijn:
a) detectie van fouten
b) localisatie van fouten
c) correctie van fouten
Bovendien zijn er combinaties van voorgaande gevallen mogelijk.

3.2.2.2. De kwaliteit van de beveiliging wordt bepaald door het maximum aantal fouten dat gedetecteerd, dan wel gecorrigeerd kan worden.

3.2.2.3. Van belang is ook de verbetering in betrouwbaarheid die verkregen wordt door het toepassen van een bepaalde beveiliging [*]. Hierbij moeten we een beoordeling geven voor de betrouwbaarheid in vergelijk met het niet beveiligde systeem, waarbij ook de feilbaarheid van de beveiliging niet uit het oog verloren mag worden.

3.2.2.4. Ook snelheid van detectie is een parameter. Indien een fout pas gedetecteerd wordt nadat de informatie reeds andere bewerkingen heeft ondergaan, dan kan door de gewijzigde machinestatus bepaalde informatie over het karakter van de fout verloren zijn gegaan.

3.2.3. De kosten. Tegenover het resultaat van de beveiliging, staan de kosten, die ervoor gemaakt moeten worden. Deze kosten zijn te splitsen in:

3.2.3.1. Kosten in de vorm van tijd.
Het doorlopen van de beveiligingslogica zal in het algemeen extra tijd kosten. Vooral is dit van belang wanneer de voortgang van de informatiestroom gekoppeld is aan een "accoord-signaal" van de beveiliging.

[*] Onder de betrouwbaarheid van een systeem verstaan we de kans dat een systeem gedurende bepaalde tijd foutloos werkt.

3.2.3.2. Kosten in de vorm van hardware. Deze kosten kunnen we nog onderverdelen in:

3.2.3.2.1. Kosten aan logica ten behoeve van bijvoorbeeld codering en decodering (aantallen nands, flip-flops).

3.2.3.2.2. Kosten die gemaakt moeten worden om een bepaalde codering toe te staan. Gaat men bijvoorbeeld over tot de beveiliging met een bepaalde code, dan zullen in ieder woord een aantal bits voor deze code gereserveerd moeten worden. Dit geldt dan voor alle woorden van het geheugen. De kosten kunnen derhalve zeer omvangrijk zijn.

3.2.4. Effect op de rest van het systeem. Het invoeren van een bepaalde beveiliging kan soms consequenties met zich meebrengen voor externe gedeelten, die we niet direct in onze beveiliging willen betrekken.

3.2.5. Vanuit commercieel oogpunt kan het van belang zijn dat een bepaalde beveiliging als optie geleverd kan worden. Indien dit niet het geval is, zullen alle klanten, ongeacht of zij de beveiliging al dan niet op prijs stellen, voor deze beveiliging moeten betalen.

In een bepaald systeem kunnen voornoemde factoren sterk van elkaar afhankelijk zijn, zodat het vaak niet mogelijk zal blijken een systeem op al deze factoren te beoordelen. We zullen dan slechts de meest relevante gegevens vermelden.

4. Mogelijkheden ter beveiliging van logische systemen.

Een van de middelen waarover we beschikken om een logisch systeem te beveiligen is het verhogen van de betrouwbaarheid.
Onder de betrouwbaarheid verstaan we de kans dat een systeem gedurende bepaalde tijd foutloos werkt.
Fouten kunnen ontstaan als gevolg van defecte transistoren, weerstanden, diodes, integrated circuits etc.
Om een hoge betrouwbaarheid te verkrijgen zullen we dus op de eerste plaats alle zorg moeten besteden aan de keuze van een juiste technologie voor onze onderdelen. We zijn hierbij zowel technisch als economisch aan grenzen gebonden. Indien daarmee de gewenste betrouwbaarheid nog niet bereikt wordt, kunnen we verbetering hiervan bewerkstelligen door het toevoegen van redundante elementen.
Onder redundante elementen verstaan we die elementen, die defecten mogen vertonen zonder dat dit van invloed is op de eigenschappen van het betreffende systeem. Met andere woorden, wanneer in een machine MO defecten optreden aan redundante elementen, dan blijft deze MO. In dat geval treedt er dus automatisch correctie op. Signalering van de aanwezigheid van een defect element vindt niet plaats. In de volgende hoofdstukken zullen we enkele voorbeelden van de verhoging van de betrouwbaarheid, door het toevoegen van redundante elementen, behandelen.
Hierbij zal worden ingegaan op:

a) verveelvoudiging van elementen
b) majority voting
c) quadded logic
d) Toepassen van codes ter beveiliging van logische systemen

Het verhogen van de betrouwbaarheid heeft alleen zin indien wij daarmee de tijdsduur tot de eerste fout in het systeem willen vergroten. Als beveiliging is deze methode ongeschikt. Na verloop van tijd zal immers ook dit betrouwbaarder systeem falen. In het geval van de machine-diagnostiek, met behulp van de ROS, zal dit dan aanleiding geven tot verkeerde beslissingen.

We nemen nu aan dat aan de betrouwbaarheid van een systeem voldoende aandacht besteed is. Dan is de meest voor de hand liggende oplossing om als beveiliging een foutdetectie in te voeren. Deze geeft géén hogere betrouwbaarheid, maar wel wordt ieder falen van

het systeem gesignaleerd. We weten dan dat er fouten gemaakt worden en kunnen hiertegen maatregelen treffen. Natuurlijk zal het ook mogelijk zijn om een foutdetectie, tegen meerkosten, uit te breiden tot localisatie of eventueel correctie. Dit laatste zal dan, naast detectie, tevens een verhoging van de betrouwbaarheid geven.
Foutdetectie kan door bepaalde coderingen verkregen worden. Er bestaat een zeer uitgebreid arsenaal aan coderingsmogelijkheden, die elk hun specifieke eigenschappen hebben. Als voorbeeld noemen we pariteit-beveiliging, Hammingcodes, cyclische codes, cross parity codes en arithmetische codes. (1)(2)(7)(9)(12)(14)

In de literatuur (4)(16)(17) zijn methoden gegeven, waarmee we, uitgaande van code-vergelijkingen, redundantie voor beveiliging kunnen toevoegen. Afhankelijk van de gebruikte code is hiermee, zowel detectie als correctie van fouten mogelijk. Hierop zal nog nader worden ingegaan.

4.1. Verveelvoudiging van elementen.

Zoals reeds vermeld vormt de toevoeging van redundante elementen een middel tot verhoging van de betrouwbaarheid. Dit kan soms op verschillende manieren geschieden. Bepaalde methoden kunnen, met een zelfde kostenfactor dan anderen toch betere resultaten geven. Aan de hand van een theoretisch voorbeeld zullen we dit nader verklaren. We maken de volgende gedachtengang. We stellen ons een systeem voor, als zijnde opgebouwd uit n elementen i, met respectievelijke betrouwbaarheid P_i. Wil het systeem correct functioneren, dan zullen alle n de elementen van het niet-redundante systeem correct moeten functioneren. De betrouwbaarheid van het systeem (fig.6) is dan:

$$P_{syst.} = \prod_{i=1}^{n} P_i$$

Om nu de betrouwbaarheid te verhogen voegen we redundante elementen toe. Stel dat we dit op twee manieren kunnen doen.

a) systeemredundantie
b) elementenredundantie

ad a) We schakelen m systemen parallel, zodanig dat het nieuwe redundante systeem goed functioneert, zolang één van de afzonderlijke systemen goed functioneert.(fig.7).

$$P_a = 1 - \left(1 - \prod_{i=1}^{n} p_i\right)^m$$

ad b) We verveelvoudigen elk der elementen i met een factor m.
Het nieuwe systeem wordt nu samengesteld uit de redundante elementen. (fig.8).
De betrouwbaarheid van dit systeem is nu:

$$P_b = \prod_{i=1}^{n} \left(1 - (1 - p_i)^m\right)$$

Ter oriëntering nemen we aan dat $p_i = p$ = constant.
Dan volgt:

$$P_{syst.} = p^n$$

$$P_a = 1 - (1-p^n)^m$$

$$P_b = (1 - (1-p)^m)^n$$

FIG. 6 (NIET-REDUNDANT)

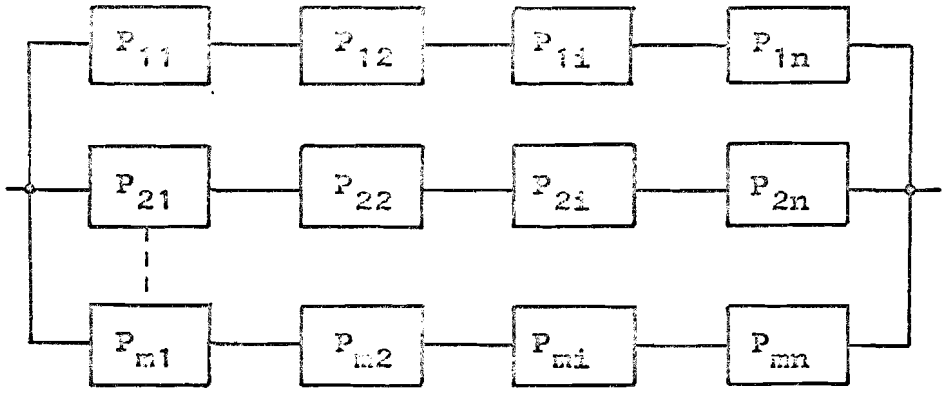


FIG. 7 (SYSTEEMREDUNDANTIE)

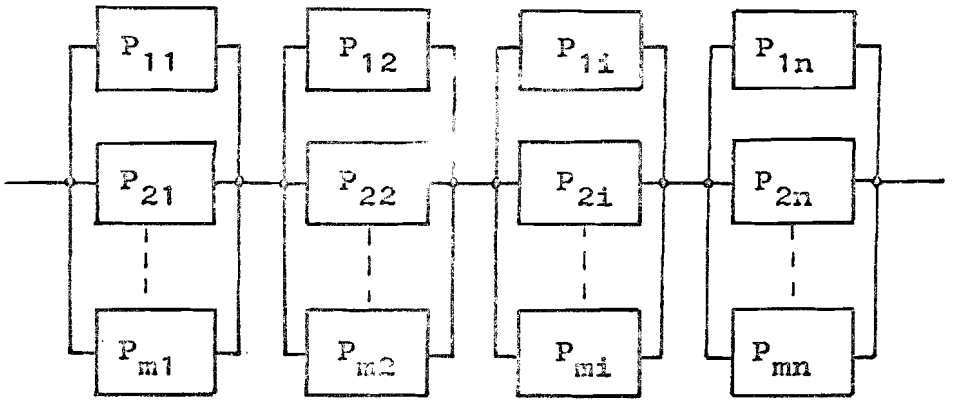


FIG. 8 (ELEMENTENREDUNDANTIE)

Vergelijken we de kosten van de systemen, dan blijkt dat systeem a even duur is als systeem b, en m maal zo duur als het niet-redundante systeem. De kosten zijn hierbij lineair met het aantal elementen gesteld.
Vergelijken we de betrouwbaarheden, dan blijkt:

$$P_b \geqslant P_a \geqslant P_{syst.}$$

Het bewijs hiervoor is in appendix I gegeven.

Uit dit theoretisch voorbeeld van redundantie concluderen we:
Redundantie kan de betrouwbaarheid verhogen.
De verhoging van de betrouwbaarheid hangt niet direct samen met de kosten die men hiervoor maakt, doch wordt in belangrijke mate mede beïnvloed door de keuze van de toegepaste methode.
Het voorgaande heeft slechts waarde als theoretische beschouwing. In het algemeen zal toevoegen van redundante elementen volgens het voorgaande schema op vele moeilijkheden stuiten. De voornaamsten hiervan worden bepaald door de koppeling van de elementen of systemen op zodanige wijze dat de redundante elementen of systemen aan het gestelde beantwoordt. De daarmee samenhangende kosten zijn zeker niet te verwaarlozen, zodat in de praktijk bovenstaande beschouwing niet volledig opgaat.

4.2. Majority voting.

We onderstellen een perfect beslissingselement, dat in staat is bits te vergelijken. De output van dit element is gelijk aan de waarde van de meerderheid van de vergeleken bits. Met dit element, we noemen het majority gate, is het mogelijk de betrouwbaarheid te verhogen en zodoende een hogere veiligheid te verkrijgen.
We beschouwen, een niet-redundant systeem S, met betrouwbaarheid r, met ingangen x_i en uitgangen y_j. De betrouwbaarheid is gedefinieerd overeenkomstig 3.2.2.3. Ter verhoging van de betrouwbaarheid kunnen we enkele van deze systemen parallel schakelen. Met behulp van een majority gate vergelijken we nu alle corresponderende y_j en bepalen hieruit y_j'. Ter verduidelijking is in fig. 9 een en ander schematisch weergegeven.
Om enkele beschouwingen over de betrouwbaarheid te houden, veron-

dan stellen we nu drie parallelgeschakelde systemen, gevolgd door een majority gate. y_j' is fout wanneer twee of meer van de systemen y_j fout zijn. Is de betrouwbaarheid van een niet-redundant systeem r en de betrouwbaarheid van het redundante systeem R, dan volgt:

$$R = r^3 + 3(1-r)r^2 = r^2(3-2r)$$

Dit is een grafiek (fig.10) afgebeeld.
We zien dat voor $r < \frac{1}{2}$ de betrouwbaarheid door toepassing van majority voting slechter wordt. Voor $r > \frac{1}{2}$ vindt een verbetering van de betrouwbaarheid plaats.

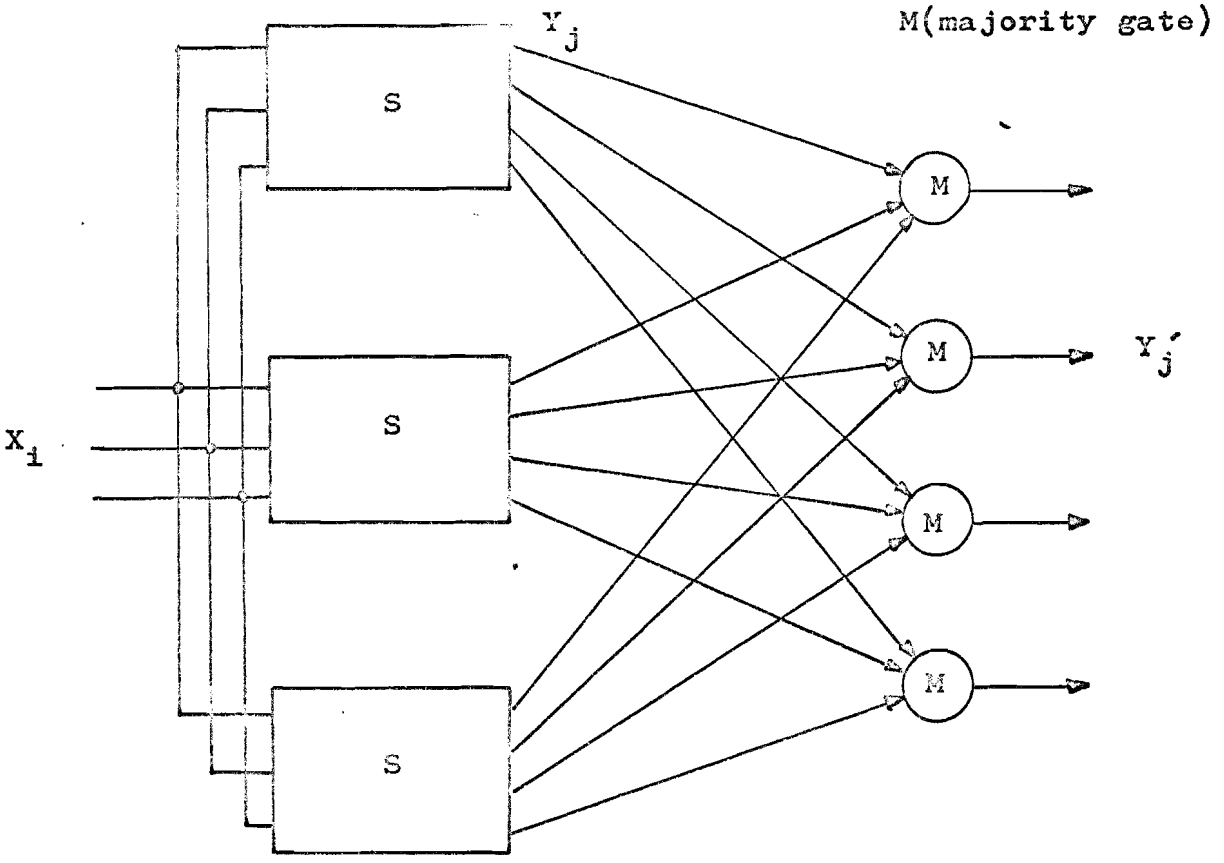


FIG. 9

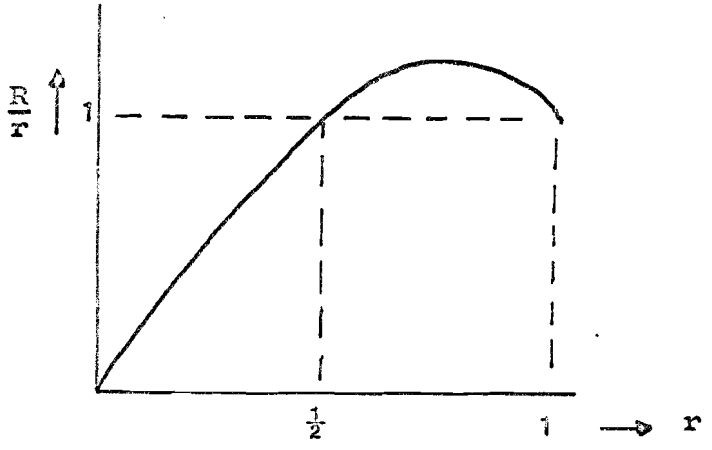


FIG. 10

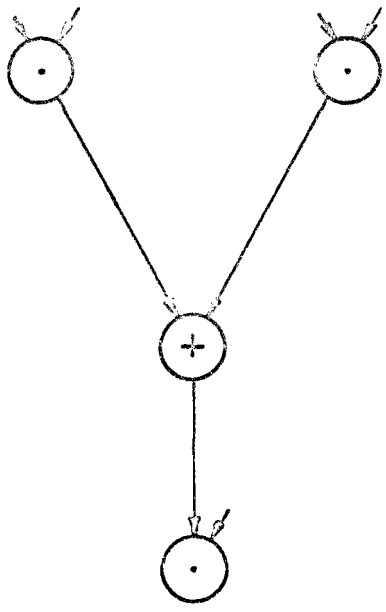
Bij het voorgaande werd geen rekening gehouden met de eindige betrouwbaarheid van de majority gate. Brengen we ook deze in rekening, dan zal de betrouwbaarheid van het redundante systeem, iets verslechteren.
Als voordeel van het op deze wijze invoeren van redundantie kan gezien worden, dat het zeer eenvoudig is een foutdetectie-inrichting aan te brengen, die dan voor signalering van fouten zorgt. Daartoe hoeven we slechts te kijken of de drie corresponderende uitgangen y_j allen gelijk zijn. Indien dit niet het geval is, signaleren we een fout. Door de aanwezigheid van de majority gate, zal het systeem toch nog foutloos kunnen doorwerken. In dit geval hebben we een gecombineerde detectie-correctiebeveiliging. Ze vergt ruim drie maal zoveel elementen als het niet-redundante systeem. (1)(3)(15)(21).

4.3. Quadded logic

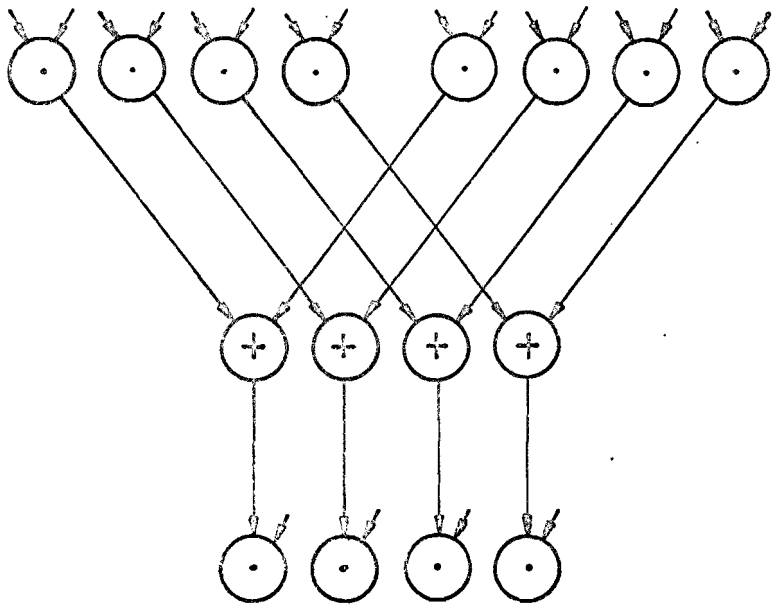
Ook quadded logic, een speciaal geval van interwoven redundant logic (3), is een toepassing van het gebruik van redundantie. Het is een methode, die er op gericht is, de betrouwbaarheid van zowel synchrone als asynchrone systemen, te verhogen. Dit geschiedt door correctie van alle enkelvoudige en de meeste meervoudige fouten.
De drie ideeën, die de basis van quadded logic vormen, zijn:

a) de logische circuits worden verviervoudigd.
b) een fout wordt automatisch, direct nadat hij veroorzaakt is, weer gecorrigeerd.
c) correctie geschiedt door de omliggende elementen van het falende element.

Om een indruk te geven van hetgeen met quaded logic wordt bedoeld, zullen we thans als voorbeeld de behandeling en foutcorrigerende werking van een eenvoudig model laten zien.
We gaan uit van een niet-redundant systeem. Dit wordt verviervoudigd. Daarna worden de nu verkregen systemen onderling verbonden, zoals in figuur 11 is aangegeven.
Is de vierde and-poort stuk, waardoor de uitgang van een 0 in een 1 verandert, dan zal deze fout zich voortplanten naar de or-poorten. Twee van deze poorten nemen nu de functiewaarde 1 in plaats van 0 aan. Ten gevolge van de corrigerende eigenschappen



a) niet-redundant



b) verviervoudigd

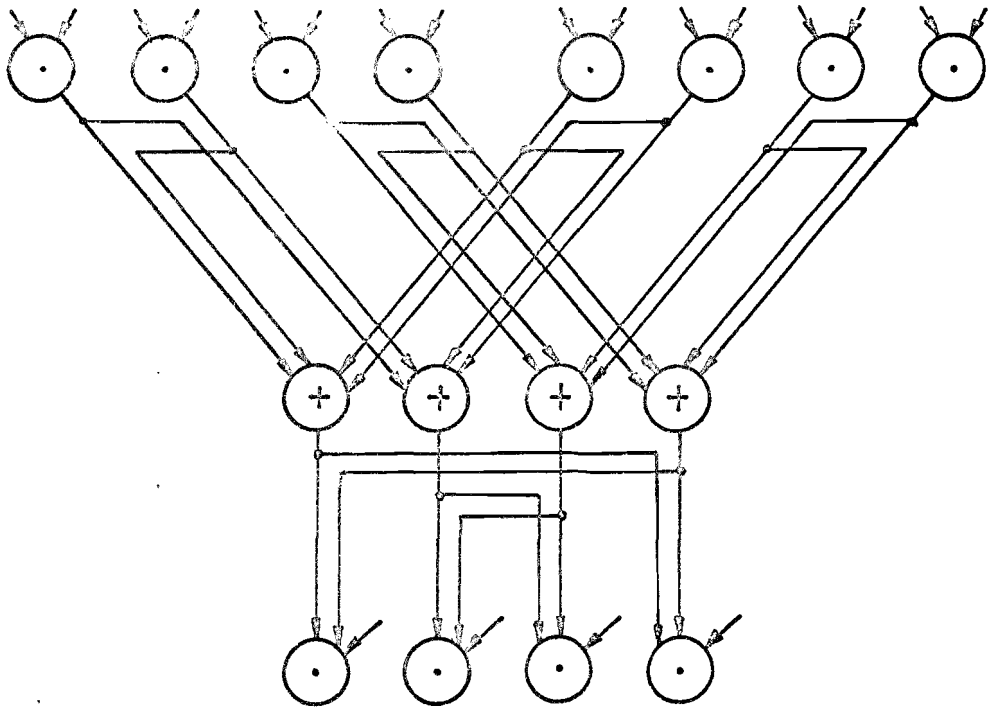


FIG. 11

c) quadded

Door de snelle correctie worden ook meerdere fouten, na elkaar ontstaan, geëlimineerd. In het beschreven voorbeeld, worden alle enkelvoudige en ongeveer 50% van de twee-voudige fouten gecorrigeerd.
Vergelijken we nu de betrouwbaarheid met het systeem zonder quaded logic, dan zien we, uitgaande van gelijke betrouwbaarheid p van de poorten;

$$r = p^4$$

$$R = \underset{\text{geen fout}}{p^{16}} + \underset{\text{één fout}}{16\,p^{15}(1-p)} + \underset{\text{twee fouten}}{0{,}50.\binom{16}{2}p^{14}(1-p)^2}$$

$$= p^{14}\left(p^2 + 16\,p(1-p) + 60(1-p)^2\right)$$

Is nu $p = 1 - d$, en $d \ll 1$ dan vinden we:

$$r = 1 - 4d$$

$$R = 1 - 60\,d^2$$

Uitgaande van voldoende betrouwbare elementen, kunnen we een aanmerkelijke verbetering van de betrouwbaarheid krijgen. Deze kan een factor 10^3 of meer bedragen.

Quaded logic technieken zijn ook toepasbaar voor sequentiele elementen. De behandeling hiervan wordt gegeven, blz. 205 tot 228.(1) Opgemerkt zij, dat er op generlei wijze een detectie van foutieve elementen plaats vindt. Dit is een zeer vervelende eigenschap van deze techniek, daar bij het optreden van fouten, de betrouwbaarheid achteruit gaat, zonder dat dit merkbaar is. Ook de localisatie van foutieve elementen wordt door de corrigerende werking sterk bemoeilijkt.
Quaded logic is duur. Zij kost ongeveer het acht-voudige van een niet-redundant systeem. De corrigerende werking is echter groot. Uit dien hoofde is zij bijzonder geschikt bij voorbeeld voor ruimtevaart-projecten. (1)(3).

4.4. Toepassen van codes ter beveiliging van logische systemen.

In het verleden zijn reeds vele bruikbare foutdetecterende en foutcorrigerende codes ontwikkeld. Met behulp van deze codes is het mogelijk een beveiliging tegen fouten in logische systemen te maken. Wanneer we een logisch systeem voorstellen als een black

ben, met n onafhankelijke ingangsvariabelen x_i en m uitgangsvariabelen y_j, dan kunnen we het verband tussen beide als volgt aangeven:

$$y_1 = y_1 (x_1 \ldots x_i \ldots x_n)$$
$$y_j = y_j (x_1 \ldots x_i \ldots x_n)$$

$$y_m = y_m (x_1 \ldots x_i \ldots x_n)$$

Het principe van deze beveiliging is nu, uitgaande van de ingangsvariabelen, een aantal functies $y_{m+1} \ldots y_{m+k}$ te construeren, zodanig dat de ontstane woorden $y_1 \ldots y_{m+k}$ deel uit te maken van een bepaalde foutdetecterende, dan wel foutcorrigerende code. Hierbij kan men als volgt te werk gaan:
met behulp van een bepaalde code bepalen we bij $y_1 \ldots y_m$, $y_{m+1} \ldots y_{m+k}$.

Daar $y_1 \ldots y_m$ functies van $x_1 \ldots x_n$, zijn ook $y_{m+1} \ldots y_{m+k}$ als functie van de onafhankelijke variabelen $x_1 \ldots x_n$ te bepalen. Naast het niet beveiligde systeem wordt daartoe een coderingsnetwerk aangebracht.(fig.12).
De uitgangssignalen $y_1 \ldots y_m$ worden samen met de toegevoegde signalen $y_{m+1} \ldots y_{m+k}$ aan een codecontrole onderworpen. Dit gebeurt in een codecontrole netwerk. Hieruit verkrijgen we, afhankelijk van de gebruikte code, foutdetecterende dan wel foutcorrigerende signalen. Willen we foutcorrectie toepassen, dan vereist dit nog een netwerk, waarin met behulp van de foutcorrigerende signalen $y_1 \ldots y_m$ gecorrigeerd worden. Bovenstaande methode is ook bruikbaar voor sequentiele netwerken. Een beschrijving hiervan wordt gegeven in 4. De foutdetecterende eigenschappen kunnen door het kiezen van een code bepaald en willekeurig hoog opgevoerd worden. In appendix 2 is als voorbeeld de beveiliging van een register met deze methode uitgewerkt.
Hieruit blijkt dat alhoewel het principe eenvoudig lijkt, voor veel toepassing de practische realisatie toch veel logica kan vergen. Een verdubbeling van het niet-redundant systeem is reeël.

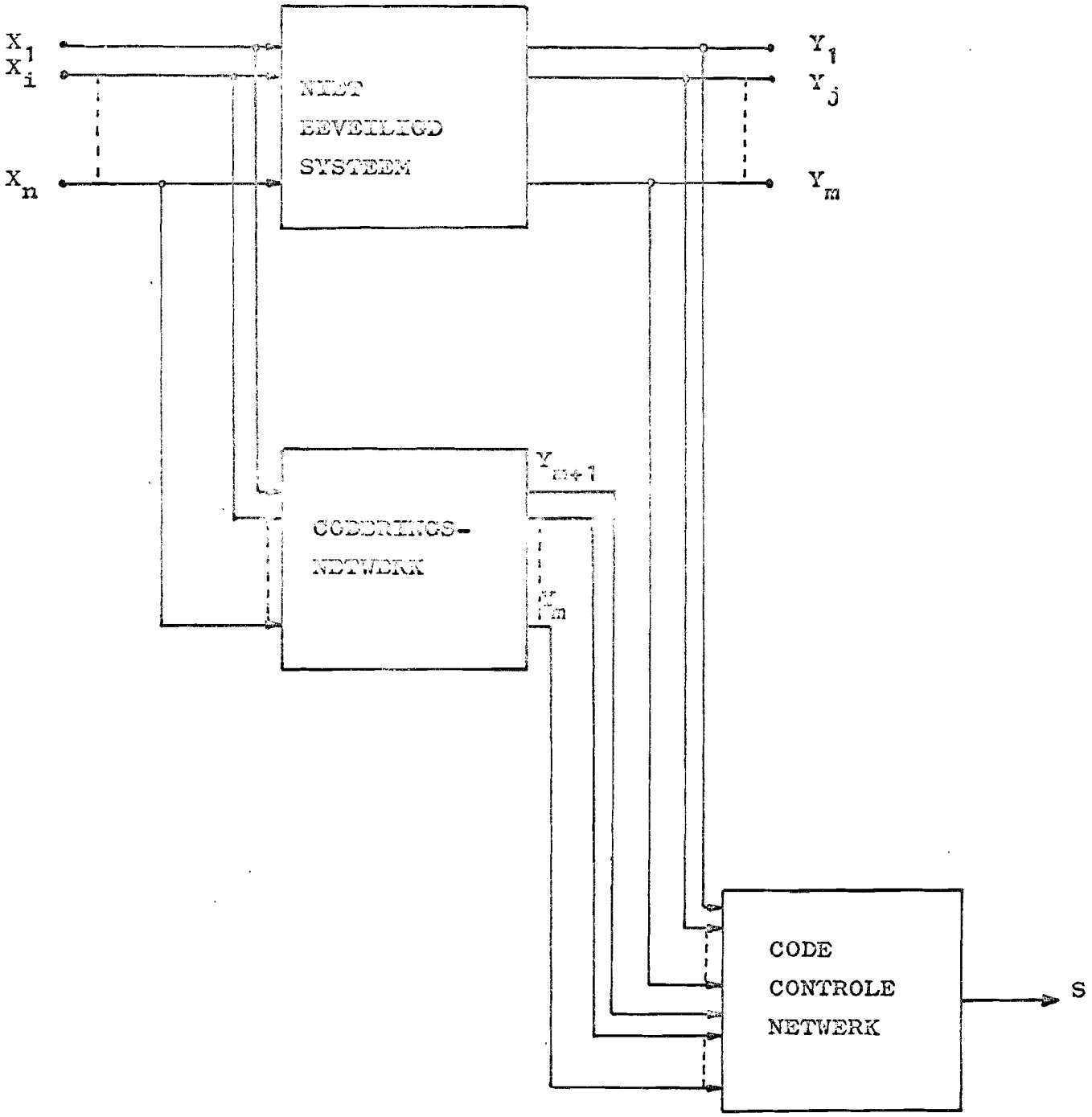

X1
Xi
Xn
NIET
BEVEILIGD
SYSTEEM
Y1
Yj
Ym
CODERINGS-
NETWERK
Ym+1
Ym
CODE
CONTROLE
NETWERK
S

FIG. 12

5. Beveiliging van de Read Only Storage.

De R.O.S., zoals deze beschreven is in hoofdstuk 2, bestaat uit een aantal logische eenheden, die elk een bepaalde taak vervullen.
Recapitulerend onderscheiden we: selectie register, selectiedecodering, geheugen, uitleesregister, commandodecodering en tijdenteller.

De functies die door de genoemde eenheden vervuld worden zijn:
I. Selectie van een nieuw adres
II. Uitlezen van het adres
III. Decodering van de inhoud
IV. Keuze van een nieuw adres
V. Register Tijden teller.

We zullen voor elk van de voornoemde functies nagaan welke mogelijkheden er zijn om een zodanig beveiliging aan te brengen, dat we zekerheid verkrijgen omtrent de juistheid van de uitgevoerde bewerking.
We zullen bij elk van deze mogelijkheden tevens aangeven hoe groot deze zekerheid is. Ook zullen we de kosten, die met het invoeren van de beveiliging gepaard gaan, aan een onderzoek onderwerpen. We maken hierbij gebruik van de beoordelingscriteria zoals deze in hoofdstuk 3.2. zijn opgesteld.

5.1. De beveiliging van de selectie.

Onder selectie verstaan we: het kiezen van het woord uit het geheugen, dat het adres draagt, zoals dit door de inhoud van het selectieregister wordt aangegeven. De beveiliging dient te controleren of dit, en slechts dit woord geselecteerd wordt. Door fouten in de selectiedecodatie kunnen andere of meerdere adressen geselecteerd worden. Dan zal de beveiliging dus een signalering van de aanwezige selectiefouten moeten geven.

Mogelijkheden die in aanmerking kunnen komen voor de beveiliging van de selectie zijn:

5.1.1. Verdubbeling van de selectiedecodatie
5.1.2. Toevoeging van een hulpdecodatie
5.1.3. Adresverdubbeling in het geheugen
5.1.4. Adrescodering in het geheugen.

5.1.1. Verdubbeling van de selectiedecodatie (fig. 13).

Naast de reeds aanwezige selectie-decodatie, wordt een tweede identieke eenheid aangebracht. Bij selectie wordt het te selecteren adres in beide eenheden op overeenkomstige wijze uitgecodeerd. Dit gebeurt bijvoorbeeld door aan dit adres een logische "1" toe te kennen. Alle andere adressen moeten dan gekenmerkt worden door een logische "0". Vergelijken we nu de overeenkomstige adres uitgangen van de twee decodatie eenheden met elkaar, dan zullen deze paarsgewijze gelijk moeten zijn. Is dit niet het geval, dan zal in een van de twee eenheden een fout zijn opgetreden. Het vergelijk kan voor elk adres op eenvoudige wijze geschieden met behulp van een exclusieve-or. Bij gelijke logische ingangswaarden zal deze een "0" als uitgang hebben. Bij verschillende ingangswaarden is de uitgang "1". Bij een foutloos werkende decodatie zullen alle exclusieve ors een "0" uitgang hebben. Een fout wordt gesignaleerd door een "1" uitgang. Door na te gaan of alle exclusieve-or uitgangen 0 zijn verkrijgen we een foutsignalering.
Dit laatste kan gebeuren met een or-schakeling. Deze zal een "0" als uitgang hebben als alle ingangen "0" zijn. Dit is dus een teken dat er geen fout aanwezig is. In alle andere gevallen zal een fout gedetecteerd worden door een "1" als uitgang van deze or-schakeling.
Door het grote aantal adressen, en daarmee samenhangend het grote aantal exlusive-or's, kunnen deze niet alle met een or vergeleken worden. Dit omdat het aantal ingangen van een or hiervoor te klein is. We kunnen echter een pyramide van or-poorten bouwen. De uitgangen van de eerste or's dienen als ingangen voor de volgenden etc. Deze, tengevolge van het te kleine aantal ingangen, noodzakelijke maatregel, opent tevens de mogelijkheid om na detectie van een fout, tot een gemakkelijke localisatie hiervan te komen.

Beveiliging.

Voor de toepasbaarheid is vereist, dat alle adreslijnen toegankelijk zijn. Deze moeten uitwendig met de beveiligingslogica verbonden kunnen worden.

De fouten in de selectiedecodatie die op deze wijze gesignaleerd worden zijn: Alle fouten die in het geselecteerde adres een "0" veroorzaken, hetgeen geen selectie zou betekenen, en alle fouten die in een of meerder van de niet geselecteerde adressen een "1" veroorzaken, hetgeen selectie van verkeerde adressen zou betekenen. M.a.w. vrijwel alle fouten worden gesignaleerd.

De beveiliging signaleert ten onrechte een fout wanneer in de beveiligingslogica een fout gemaakt wordt. De kans hierop is even groot als de kans op een foutieve adresselectie van een bepaald bit. Deze is dus verwaarloosbaar. In de praktijk is deze kans kleiner dan 10^{-40}. Ook kan een detektiefout optreden wanneer de exclusive-or's of de or-poort defecten vertoont. Door geschikte dimensionering en onderdelen keuze is deze kans voldoende klein te krijgen. Mocht dit echter niet voldoende zijn, dan kan ook hier tot verdubbeling overgegaan worden.

Een selectiefout wordt met behulp van deze beveiliging zeer snel gedetecteerd. Na de adres vorming leveren alleen de exclusieve-or's en de or-schakeling enige vertraging op. Wanneer de vertragingstijd van een poort τ is, dan zal de totale vertragingstijd ongeveer 5 τ bedragen. Door direct na de signalering van een fout, de machinestatus niet meer te wijzigen, wordt informatie over de fout bewaard. Hiermee kan op eenvoudige wijze foutlocalisatie tot stand gebracht worden, omdat in het algemeen iedere fout in de selectiedecodatie een bepaald foutenpatroon aan de adreslijnen tot gevolg zal hebben.

Bij beveiliging op deze wijze is aan de oorspronkelijke constructie niets gewijzigd. Hierdoor is ook de snelheid van de R.O.S. niet beïnvloed.

De hardwarekosten zijn gehoel in de vorm van kosten aan logica. Deze zijn echter zeer omvangrijk. Benodigd zijn: een volledige selectiedecodatie. Voor elk adres een exclusive-or, een aantal or-poorten afhankelijk van het aantal ingangen van deze poorten. Wanneer we de kosten van de bouwstenen uitdrukken in eenheden e, en we rekenen voor de selectie-decodatie 5000 e, voor een exclusive-or 5 e en voor de or-schakeling een aantal e gelijk aan het aantal ingangen gedeeld door aantal ingangen bij een ROS met 20.000 adressen 107.200 e. Het aantal ingangen per poort is hierbij 10 verondersteld.

De beveiliging is zodanig uit te voeren, dat levering als optie mogelijk is.

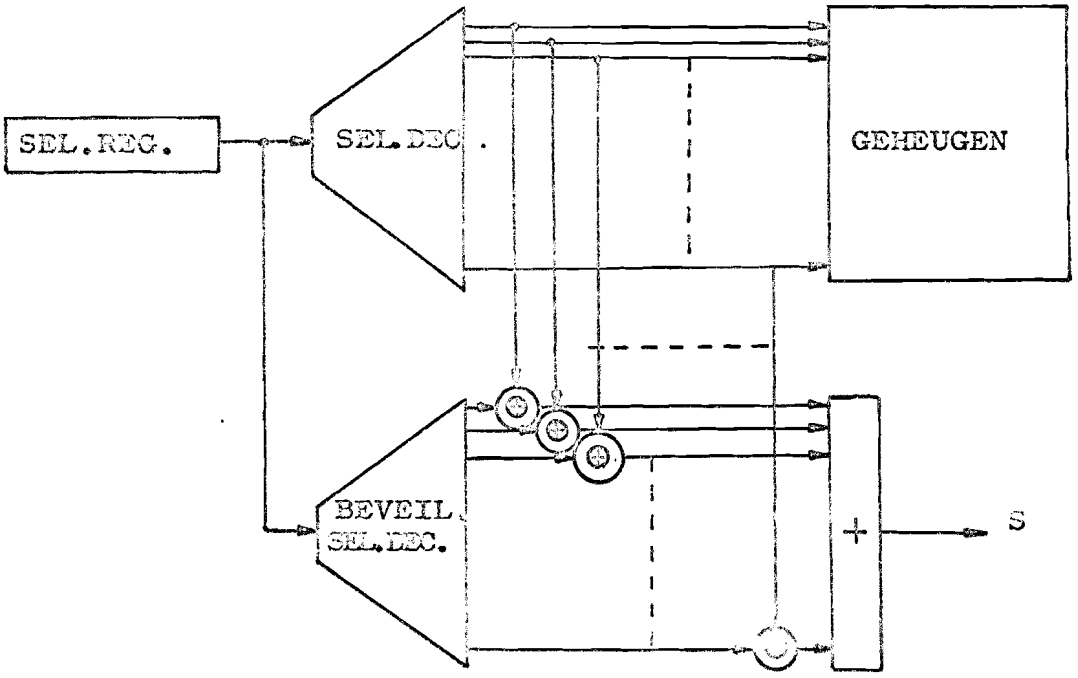

SEL.REG.
SEL.DEC.
GEHEUGEN
BEVEIL
SEL.DEC.
+
S

FIG. 13

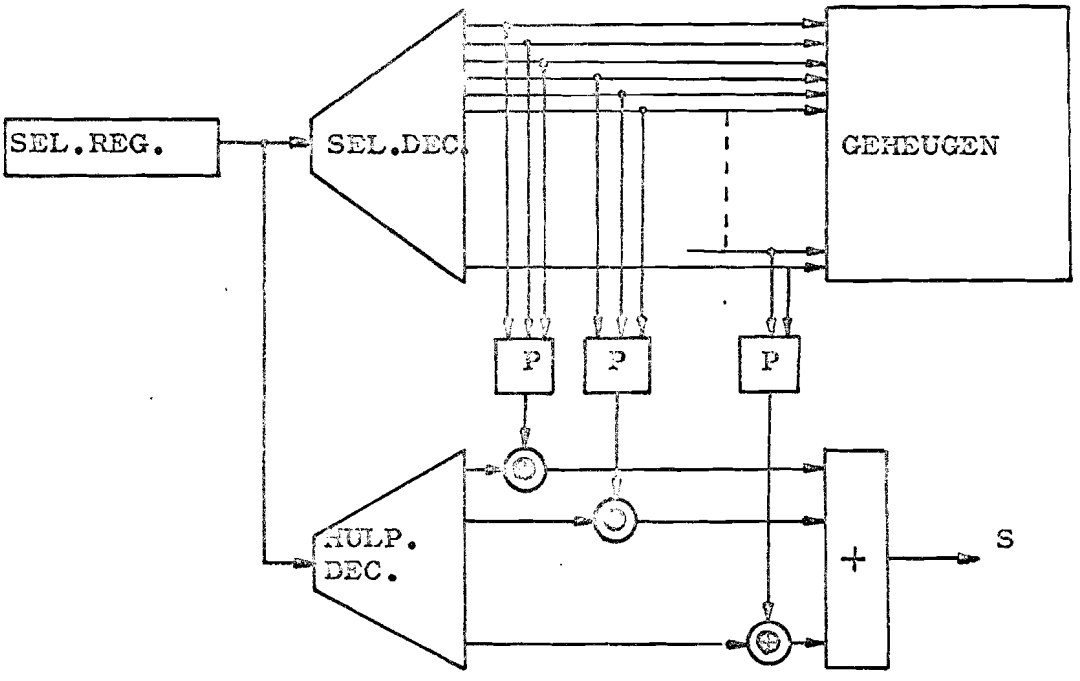

SEL.REG.
SEL.DEC.
GEHEUGEN
P
P
P
HULP.
DEC.
+
S

FIG. 14

5.1.2. Toevoegen van een hulpdecodatie. (fig. 14).

Als variant op de voorgaande beveiliging kunnen we de adreslijnen in bepaalde groepen indelen. In elk van deze groepen gaan we na of de pariteit "1" dan wel "0" is. Met een hulpdecodatie bepalen we de pariteit, zoals deze voor elke groep zou moeten zijn, rechtstreeks uit de inhoud van het selectieregister. Voor de groep die het te selecteren adres bevat is de pariteit "1". Voor alle anderen "0". Met exclusive-or's vergelijken we op overeenkomstige wijze als bij 5.1.1. de uitgangen van de pariteit-schakelingen met de uitgangen van de hulp decodatie.
De foutsignalering wordt verkregen door de exclusive-or's met een or-schakeling te verbinden.

Beoordeling.

De beoordeling komt in grote lijnen overeen met de beoordeling van 5.1.1. Wij zullen slechts op de verschillen tussen de beveiligingen ingaan.

De kwaliteit van de beveiliging is lager dan die van de voorgaande. Terwijl de voorgaande beveiliging vrijwel alle mogelijke selectiefouten detecteerde, detekteert deze beveiliging één selectie-fout per groep. De detektiecapaciteit hangt sterk af van het aantal groepen. Als we voor iedere vijf adressen één groep nemen, dan worden per groep 50 % van alle mogelijke fouten gedetecteerd. De beveiliging detecteert echter wel alle enkelvoudige fouten binnen een groep. Aangezien de kans op een tweevoudige fout binnen een groep zeer klein is, zal deze beveiliging ook zeer goede resultaten geven.

De detectiesnelheid is ongeveer gelijk aan die van beveiliging 5.1.1. Voert men echter de pariteitschakeling uit als serie-teller, hetgeen goedkoper kan zijn, dan wordt de detectiesnelheid aanmerkelijk lager.
De voornaamste redenen die tot de keuze van deze beveiliging, in vergelijk met 5.1.1., kan leiden zijn de lagere kosten. De hulpdecodatie kan aanmerkelijk eenvoudiger zijn dan de in 5.1.1

benodigde selectiedecodatie. We kunnen volstaan met een decodatie van 1000 e. Voor de pariteitschakelingen hebben we per groep 16 e nodig. Het aantal exclusive-or's is gereduceerd. Ook de or-schakeling is eenvoudiger geworden. In totaal hebben we nodig voor de selectie beveiliging met deze methode 85.400 e. Dit is ongeveer 20% minder dan bij methode 5.1.1.

5.1.3. Adresverdubbeling in het geheugen (fig. 15).

Een andere mogelijkheid om de selectieprocedure te beveiligen verkrijgen we door bij ieder woord het eigen adres op te nemen in het geheugen. Bij het lezen van een woord, na de selectie, verschijnt dan het adres in het uitleesregister. Dit kan dan vergeleken worden met het adres in het selectieregister. Met behulp van exclusive-or's kan dit vergelijk op eenvoudige wijze plaats vinden. Een foutsignalering krijgen we door de uitgangen van de exclusive-or's toe te voegen aan een or-schakeling.

Beoordeling.

De toepasbaarheid van deze beveiliging hangt af van: het al dan niet beschikbaar zijn van voldoende geheugenruimte.

De beveiliging geeft een detectie van alle fouten die de selectie van een verkeerd adres tot gevolg hebben. Indien ten gevolge van fouten, meerdere adressen tegelijk geselecteerd worden, is detectie hiervan niet verzekerd, doch wel zeer waarschijnlijk. Door het or-en van de informatie is de kans dat een "0" in een "1" verandert zeer groot. Detectie vindt dan ook in dit geval vrij zeker plaats. Indien het adres veel "0"-en bevat is de kans op detectie groter dan wanneer het adres is opgebouwd uit veel "1"-en.
Localisatie van fouten kan vergemakkelijkt worden door een vergelijk tussen de inhouden van het selectieregister en het uitleesregister.

De betrouwbaarheid van de R.O.S. wordt door het aanbrengen van de beveiliging niet verhoogd. De betrouwbaarheid van de beveiliging is door de geringe hoeveelheid aan benodigde logica erg groot.

De detectie van selectiefouten vindt snel plaats. De enige vertraging die hierbij een rol speelt wordt bepaald door de schakelsnelheid van de exclusive-or's in het vergelijkingsorgaan, zodat na het uitlezen de vertraging ongeveer 3 τ is.

De beveiliging kan een kleine vertraging van de R.O.S. tot gevolg hebben. Het inschrijven van het volgende adres in het selectieregister kan pas geschieden nadat het vergelijk betreffende het geselecteerde adres heeft plaats gevonden. Als de hierdoor ontstane vertraging te groot wordt, kan de inhoud van het selectieregister tijdig overgebracht worden naar een bufferregister. Het vergelijk kan dan in dit register plaats vinden en het nieuwe adres kan direct naar het selectieregister worden overgebracht.

De kosten van de beveiliging op deze wijze zijn tweeledig. Ze bestaan enerzijds uit kosten in de vorm van logica ten behoeve van het vergelijk tussen uitleesregister en selectieregister. Voor 15 selectiebits zijn deze kosten 77 e. Anderzijds bestaan de kosten uit geheugen ruimte. Wanneer we voor een woord met 120 bits de kosten gelijk aan e stellen, dan zijn de totale kosten voor de reservering van 15 bits per woord $\frac{15}{120}$ x 20000 e = 2500 e. De totale kosten van deze beveiliging zijn dan 2577 e. Gezien de kwaliteit van deze beveiliging, die slechts weinig minder is dan 5.1.1. is deze beveiliging, ook relatief, veel goedkoper.

Deze methode van beveiliging is moeilijk als optie uit te voeren. Ze eist een aanzienlijk deel van het geheugen op, zodat bij het schrijven van de microprogramma's hiermee rekening gehouden moet worden. Dit kan dan gevolgen hebben voor het ontwerp van het deel van de machine buiten de R.O.S.

5.1.4. Adrescodering in het geheugen (fig. 16).

Aan de beveiliging zoals deze in 5.1.B beschreven zijn:
De beveiliging eist te veel geheugenruimte op, en de beveiliging is niet afdoende tegen selectiefouten waarbij meerdere adressen tegelijk worden uitgelezen.

De hoeveelheid geheugenruimte die door de beveiliging in beslag wordt genomen kan gereduceerd worden door in plaats van het gehele adres, slechts een codewoord voor dit adres in het geheugen op te nemen. Voor vergelijk dienen we dan ook dit codewoord uit de adres informatie in het selectieregister te vormen. Hiertoe maken we gebruik van een coderingsnetwerk. Op dezelfde manier als in 5.1.3. wordt nu de inhoud van het uitleesregister vergeleken met de uitgang van het coderingsnetwerk.

Een beveiliging tegen het uitlezen van meerdere adressen tegelijk verkrijgt men door naast de opname van het adres of codewoord in het geheugen, tevens het inverse van dit adres of codewoord in het geheugen op te nemen. Indien nu twee woorden, die niet door hetzelfde adres of codewoord gekenmerkt zijn, tegelijk worden uitgelezen, dan zal dit altijd aanleiding geven tot verandering van het adres, respectievelijk codewoord of inverse hiervan.

Een andere mogelijkheid om bij selectie van meerdere adressen tegelijk, detectie waarschijnlijker te maken is het opnemen van een pariteitbit bij het adres codewoord. Dit kan eventueel als deel van de uitleesbeveiliging geschieden (zie 5.2.).

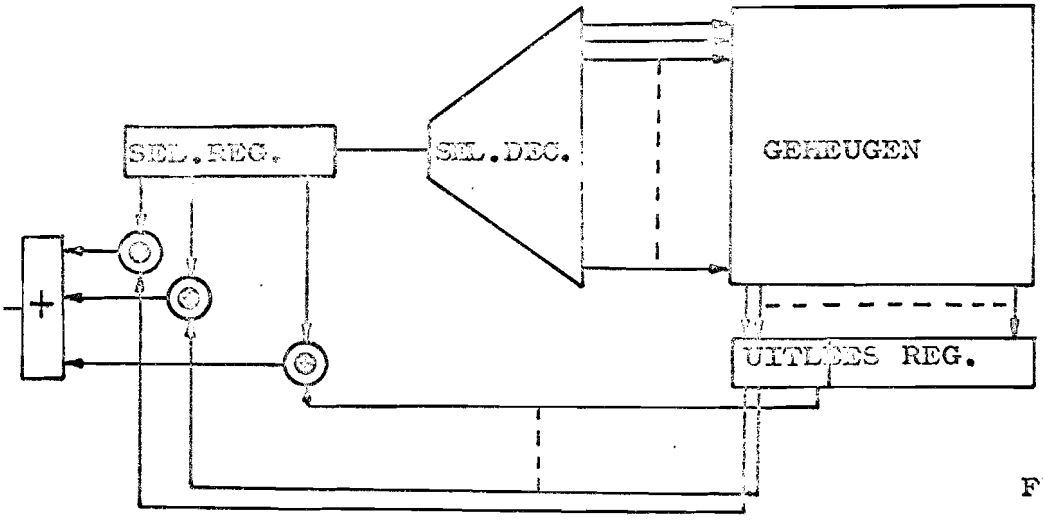

SEL.REG.
SEL.DEC.
GEHEUGEN
UITLEES REG.

FIG. 15

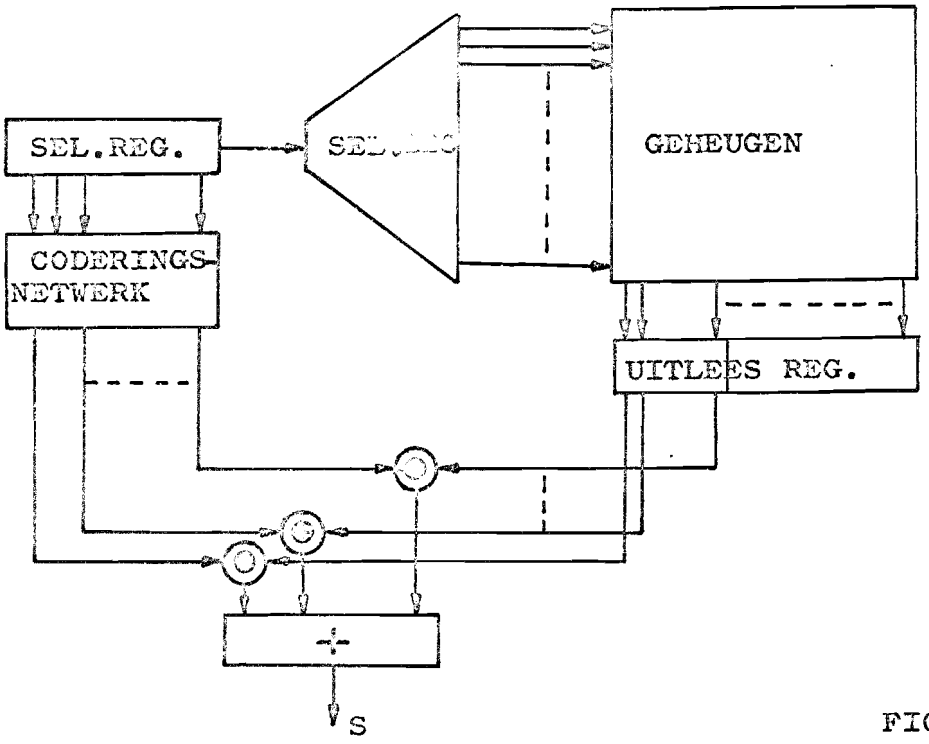

SEL.REG.
GEHEUGEN
CODERINGS-
NETWERK
UITLEES REG.
S

FIG. 16

Beoordeling.

De beoordeling komt in grote lijnen overeen met de beoordeling van beveiliging 5.1.3.
De verschillen, die er tussen beide oplossingen zijn, zijn gelegen in de foutdetektiemogelijkheden, en de benodigde geheugenruimte.
De benodigde geheugenruimte is afhankelijk van het aantal bits van het adres codewoord. De adrescode bepaalt de detectiemogelijkheid voor de selectiebeveiliging.
Bij de keuze van een adrescode dienen we ons te laten leiden door gegevens betreffende de foutfrequentie en de soort fouten die optreden. (onafhankelijk, afhankelijk). Selectie van meerdere adressen gelijktijdig is ook een van de fouten, die met de reeds aangegeven methode gedetekteerd kan worden.

Codes die voor toepassing in aanmerking komen zijn velerlei. We geven hier enkele voorbeelden. Enkelvoudige Pariteit is ongetwijfeld de goedkoopste. Ze kost slechts een geheugenbit per woord. Ze geeft een detectie van alle enkelvoudige fouten. De beveiliging met een enkelvoudige pariteit is effectief indien de kans op een foutief onderdeel vrij klein is, en bovendien zo'n onderdeel slechts een enkelvoudige fout kan veroorzaken. Kan door het defect raken van een onderdeel selectie van twee adressen optreden, dan kan uitbreiden van het pariteitbit met één inverse bit een oplossing bieden, die detektie van alle enkelvoudige fouten waarborgt.

Is de foutkans groter, zodat enkelvoudige pariteit beveiliging geen oplossing biedt, maar blijven de fouten onafhankelijk, dan kunnen het beste pariteitbits voor delen van het adreswoord worden opgenomen. De beveiliging voldoet nu zolang de kans op een meervoudige fout in een van de delen van het adreswoord voldoende laag is.
Een code die alle tweevoudige fouten detecteert is de enkelvoudige Hamming Code.
Ook cyclische codes kunnen toegepast worden ter detectie van fouten. Ze zijn bijzonder geschikt indien er voldoende tijd is

om het codewoord in serie te vormen. In het algemeen zal dit echter weinig voorkomen daar de kosten van de coderingslogica klein zijn ten opzichte van de kosten van de gereserveerde geheugeruimte.

Veelal worden door verkeerde selectie een aantal woorden uitgelezen die geen enkele binding met elkaar hebben. Theorieën over enkel- of meervoudige fouten gaan dan niet op, omdat er geen simpele relatie tussen de gekozen woorden en het adres in het selectieregister bestaat. Toch kan hier beveiliging met één of meerdere pariteitsbit een detectie geven. Naarmate meer pariteitbits gebruikt zijn is de kans dat een woord uit een "verboden" klasse gekozen wordt groter. Voor een bit is deze kans $\frac{1}{2}$. Voor twee bits $\frac{3}{4}$, voor 3 bits $\frac{7}{8}$ etc. Detectie vindt dan altijd plaats, wanneer bij het codewoord ook het inverse woord is opgenomen.
Bij het gebruik van een pariteitbit plus inverse bit vindt, bij uitlezen van 8 woorden gelijktijdig, in 99,6% $(1 - \binom{8}{0}(\frac{1}{2})^0(\frac{1}{2})^8$ van de gevallen detectieplaats. Hierbij is verondersteld dat de geselecteerde woorden onafhankelijk zijn, en dat bij uitlezen de informatie van alle gelezen woorden ge-ored is.

De kosten van de te reserveren geheugenruimte zijn in dit geval 20.000 x $\frac{2}{125}$ e = 320 e. De kosten van de decoderingslogica zijn ongeveer 60 e, zodat de totale beveiliging ongeveer 400 e kost.

De vertragingstijd bij deze beveiliging wordt veroorzaakt door het vergelijk tussen decodering en adrescode. Ze bedraagt ongeveer 2 τ .

Resumerend zien we dat de beveiliging van de selectie door middel van verdubbeling van de selectiedecodatie 5.1.1. toegankelijkheid tot de adreslijnen vereist. Deze beveiliging kost bovendien zeer veel logica. Een besparing hierop kan worden verkregen methode 5.1.2. toe te passen, waarbij een veel eenvoudigere hulpdecodatie kan worden gebruikt.
Adresverdubbeling 5.1.3. daarentegen kost zeer weinig logica, maar veel geheugenruimte. Door een geschikte codering te gebruiken in plaats van het gehele adres, kan ook hiervan veel beperkt worden. 5.1.4.

Een algemeen voordeel van de beveiliging d.m.v. adres verdubbeling in het geheugen ten opzichte van decodatie verdubbeling is, dat de eerst genoemde tevens reeds een controle op de uitleesprocedure bevat. Bovendien is de kans op het falen van de beveiligingslogica minder groot, omdat deze uit veel minder onderdelen is opgebouwd.

5.2. Toepasbaarheid van de uitleesprocedure.

Uitlezen van de R.O.S. geschiedt door een bepaald adres met een leesstroom te bekrachtigen. Het gehele woord, behorende bij dit adres wordt dan gelijktijdig gelezen. Alle bits worden parallel opgevangen en in het uitleesregister geplaatst. Indien er een fout ontstaat in de leesstroom, dan zal dit in een groot aantal bits merkbaar zijn. Door een aantal pariteitsbits zal een dergelijke fout vrijwel steeds gedetecteerd worden. In het bijzonder, wanneer we de selectiebeveiliging 5.1.3. of 5.1.4. nemen zal detectie van een dergelijke fout optreden.
Is echter de leesstroom correct, dan zullen fouten slechts optreden in fysische verbindingen en onderdelen zoals flip-flops. Dit soort fouten is vrijwel steeds onafhankelijk van elkaar. Daarom kunnen we hier ter beveiliging pariteitbits gebruiken. Daartoe delen we het woord op in een aantal groepen, zodanig dat de kans op het optreden van twee of meer fouten tegelijk, binnen een groep, zeer klein is. Voor elk van deze groepen nemen we een pariteitbit en plaatsen dit in het geheugen. Na het uitlezen van een woord, wordt de pariteit gecontroleerd. Beveiligen met pariteit zal in dit geval, wanneer de fouten onafhankelijk zijn, de meest effectieve manier van beveiligen zijn. Ook andere codes kunnen hier worden toegepast. (zie 5.1.4.)

Beoordeling.

De toepasbaarheid van de beveiliging is alleen afhankelijk van de beschikbaarheid van voldoende geheugenruimte.
De beveiliging is gericht op detectie van fouten.

De kwaliteit van de beveiliging is afhankelijk van de gebruikte code. Zo kan willekeurig hoog opgevoerd worden. Bij beveiliging met pariteit, vindt in iedere groep, waarvoor een pariteit bit ge-

bruikt is, enkelvoudige foutdetectie plaats.

Ook zal, wanneer de adresselectie foutieve adressen kan selecteren, die niet afhankelijk zijn van de inhoud van het selectieregister (zie 5.1.4.), deze beveiliging een extra detectie mogelijkheid voor selectiefouten zijn. Ook hier kunnen om deze bij de pariteitbits tevens de inverse bits worden opgenomen.

De betrouwbaarheid van de beveiliging is zeer goed. Ze hangt sament met de betrouwbaarheid van de pariteitschakelingen, die gebruikt worden.

Detectie van fouten vindt snel plaats. De vertragingstijd is afhankelijk van de logica, die voor de pariteit-controle gebruikt wordt. Ze bedraagt ongeveer 4 τ .
Er is tevens een gedeeltelijke localisatie, omdat de pariteit de groep aangeeft waarbinnen de fout gemaakt.

De kosten die voor de beveiliging gemaakt moeten worden bestaan uit twee gedeelten. We moeten geheugenruimte reserveren voor de pariteitbits. Bovendien is een hoeveelheid logica nodig voor de pariteit controle. Gebruiken we bijvoorbeeld vier bits, dan zijn de kosten van de te reserveren geheugenruimte $\frac{4}{120}$ x 20000e = 667 e. Voor de pariteit controle hebben we ongeveer 400 e nodig. De totale kosten zijn dus ongeveer 1100 e.

De R.O.S. ondervindt geen vertraging door het toepassen van deze beveiliging.

De beveiliging brengt geen consequenties voor de rest van de systeemopbouw met zich mee.

Levering als optie is moeilijk te verwezenlijken. (Zie opmerkingen bij 5.1.3.)

5.3. Beveiliging van de commandodecodering

De commandodecodering, zoals deze beschreven is in hoofdstuk 2, krijgt haar informatie uit het uitleesregister. Ze dient, afhankelijk van de inhoud van het uitleesregister, een aantal lijnen te bekrachtigen. De beveiliging heeft tot taak na te gaan of

uitgaande van correcte informatie in het uitleesregister, de juiste lijnen bekrachtigd worden. Indien dit niet het geval is moet foutsignalering plaatsvinden.
Commandolijnen,die nooit gelijktijdig bekrachtigd worden,zijn in een veld ondergebracht. Zo ontstaan een aantal velden, die elk een aantal bits van het uitleesregister omvatten en die elk afhankelijk van de waarden van deze bits maximaal één commandolijn bekrachtigen. De velden zijn verder onafhankelijk van elkaar. We zullen daarom de beveiligingsmogelijkheden voor een veld nagaan. Deze methode kan dan,verveelvoudigd,op alle velden worden toegepast.
We merken op dat een veld qua structuur grote analogie vertoont met de selectie-decodatie. Bij beide decodaties wordt, uitgaande van de informatie in een register, uit een aantal lijnen een bepaalde lijn geselecteerd. Er is echter wel een aanmerkelijk verschil in de grootte van de genoemde decodaties. Bij het selectie register hebben we een register met een inhoud van 10 à 16 bits en dus 1000 à 50000 lijnen. Bij een veld is dit 5 à 16 lijnen. Voor de beveiliging kan geen gebruik gemaakt worden van selectie van geheugen adressen, zodat oplossingen analoog aan 5.1.3 en 5.1.4 niet mogelijk zijn.
Mogelijkheden die in aanmerkingkomen ter beveiliging van de commando-decodatie zijn:
5.3.1. Verdubbeling van de commando-decodatie
5.3.2. Pariteitcontrôle
5.3.3. Terugcodering
5.3.4. Modulo 3 codering

5.3.1. Verdubbeling van de commando-decodatie (fig.17).

Vanwege de analogie tussen de selectiedecodatie en de commando-decodatie ligt het voor de hand oplossingen die we voorstelden ter beveiliging van de selectiedecodatie ook hier op hun bruikbaarheid te toetsen. De eerste oplossing is het verdubbelen van de commando-decodatie.
Hiertoe wordt, in overeenstemming met oplossing 5.1.1., naast de commando-decodatie een tweede identieke decodatie aangebracht. De corresponderende uitgangen van de decodaties worden

met behulp van exclusive-or's vergeleken. De uitgangen van de exclusive-or's worden ge-ored. Indien een fout in een van de decodaties is, zal de or-uitgang dit signaleren met een logische "1". Een meer gedetailleerde beschrijving hiervan is gegeven bij de selectiedecodatie 5.1.1.

Beoordeling.

Ook betreffende de beoordeling komen veel punten overeen met 5.1.1. Het effect van de beveiliging is detectie van fouten. De beveiliging detecteert vrijwel alle fouten.

De betrouwbaarheid van de beveiligheid is goed, daar de kans dat in beide decodaties gelijktijdig dezelfde fout ontstaat zeer klein is. Ook indien de beveiligingsdecodatie faalt, zal dit gesignaleerd worden. Deze signalering is niet te onderscheiden van die bij een werkelijke fout in de commando-decodatie.

De detectie van fouten vindt snel plaats. De vertragingstijd is 4 τ . Hierdoor kan gemakkelijk nagegaan worden waar een eventuele fout gemaakt is.

Door het aanbrengen van de beveiliging wordt de R.O.S. niet vertraagd, omdat de oorspronkelijke decodatieopbouw niet gewijzigd wordt.

De kosten van de beveiliging zijn hoog. Voor beveiliging zijn nodig een volledige commando-decodatie, een aantal exclusive-or's gelijk aan het aantal commandolijnen, en bovendien een aantal or-poorten die voor de uiteindelijke foutsignalering zorgen. Drukken we de kosten uit in eenheden, dan zijn deze voor een veld van 4 bits, en 16 commandolijnen gelijk aan 98 e. Voor de gehele decodatie, bestaande uit 25 velden, zijn de kosten dan ongeveer 2500e.

De beveiliging kan zodanig geconstrueerd worden dat levering als optie mogelijk is.

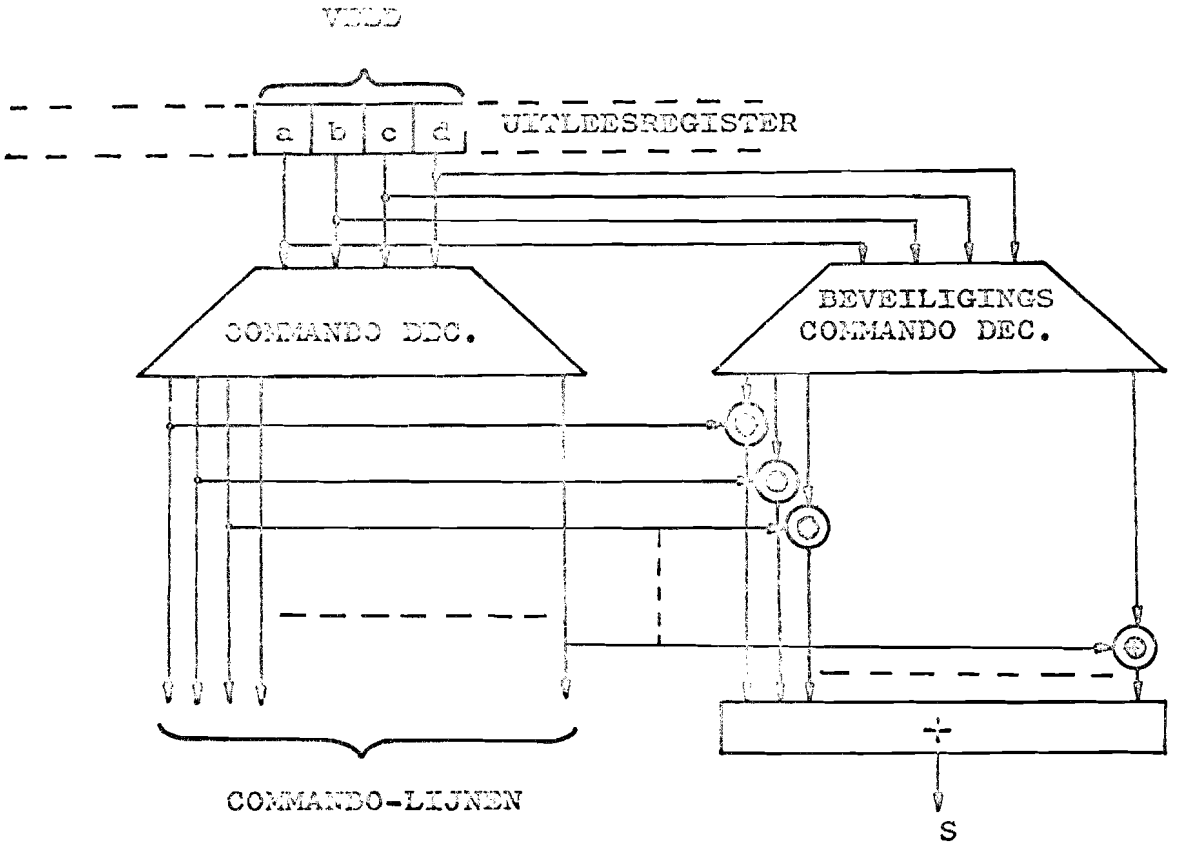

a
b
c
d
UITLEESREGISTER
COMMANDO DEC.
BEVEILIGINGS
COMMANDO DEC.
COMMANDO-LIJNEN
S

FIG. 17

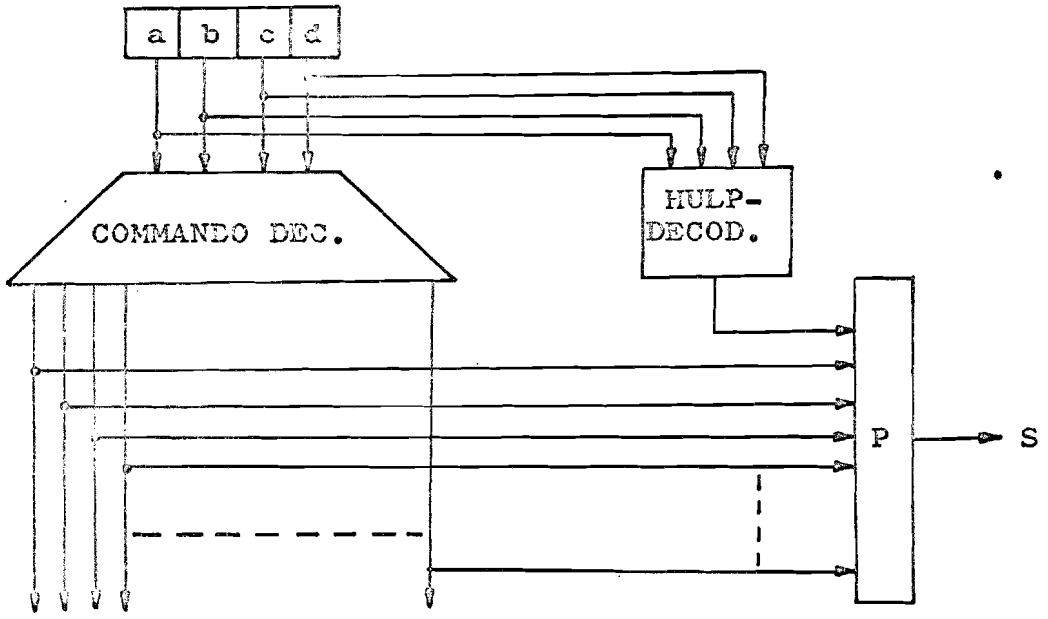

a
b
c
d
COMMANDO DEC.
HULP-
DECOD.
P
S

FIG. 18

5.3.2. Pariteit contrôle (fig. 18)

Een opmerkelijke eigenschap van de commando-decodatie is dat in elk veld nooit meer dan een lijn geselecteerd mag worden. Beschouwen we nogmaals de opbouw, dan zien we dat elk veld uit een aantal bits bestaat. Deze zorgen via poorten voor de bekrachtiging van de commandolijnen. Bij elke toestand van de veldbits wordt één of géén commandolijn geselecteerd. Wanneer we nu alle toestanden waarbij geen commandolijn geselecteerd wordt naar een lijn uitcoderen, zodanig dat in die toestanden deze lijn geselecteerd wordt, dan zal in ieder veld steeds één lijn geselecteerd zijn. De informatie op de lijnen vormt dan een één uit n code, waarbij n het aantal lijnen in een veld is.
De beveiliging kan nu op eenvoudige wijze aangebracht worden door pariteit van het veld te bepalen. Bij een goed functionerend veld zal deze pariteit steeds één moeten zijn. In alle andere gevallen moet foutsignalering plaats vinden.

Beoordeling.

De beveiliging detecteert alle enkelvoudige fouten. Een fout ontstaat meestal door het defect raken van een poort. Deze neemt dan permanent de waarde 0 of 1 aan. De poorten zijn onafhankelijk van elkaar. Indien bovendien de foutkans binnen een veld voldoende laag is, zodat het gelijktijdig defect raken van meerdere poorten zeer onwaarschijnlijk is, dan zullen de veroorzaakte fouten enkelvoudig zijn.

De betrouwbaarheid van de beveiliging is afhankelijk van de betrouwbaarheid van de pariteitschakelingen. Deze is echter zeer goed.

De signalering van fouten vindt ook bij deze beveiliging snel plaats. De enige vertraging treedt op in de pariteitschakeling. Deze zal 3 τ zijn.

De kosten van de beveiliging zijn een aantal pariteitschakelingen gelijk aan het aantal velden, en een or met een gelijk aantal ingangen.

Voor 25 velden van 4 bits zijn deze kosten ongeveer 1250e.

Levering als optie is ook bij deze beveiliging goed mogelijk.

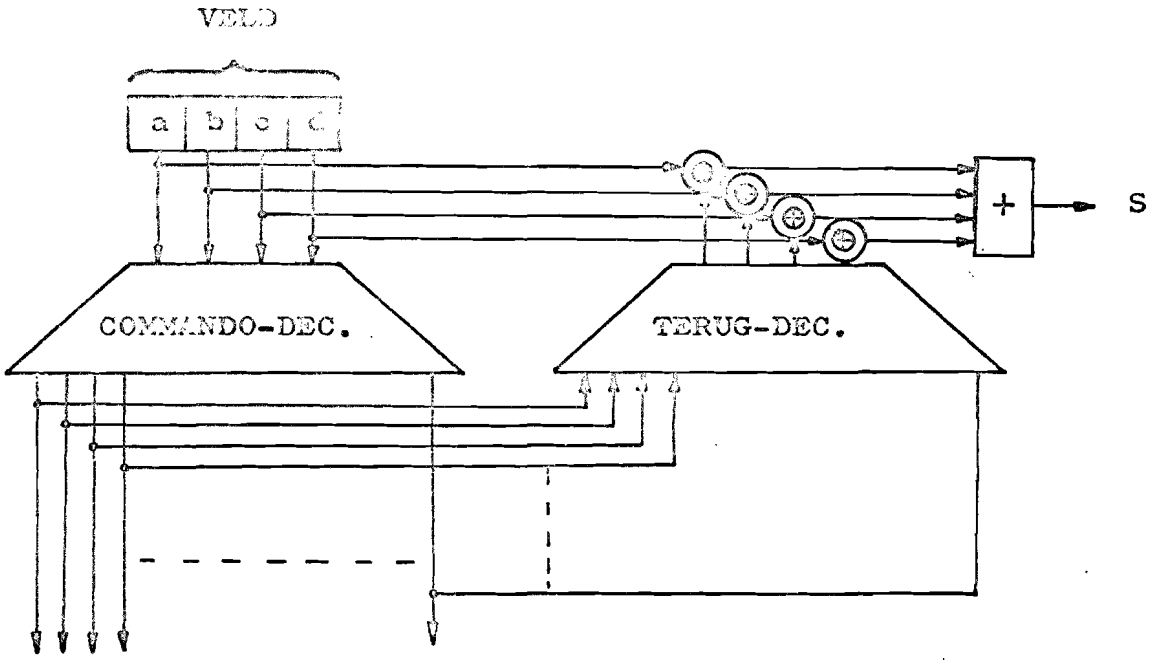

VELD
a
b
c
d
+
S
COMMANDO-DEC.
TERUG-DEC.

FIG. 19

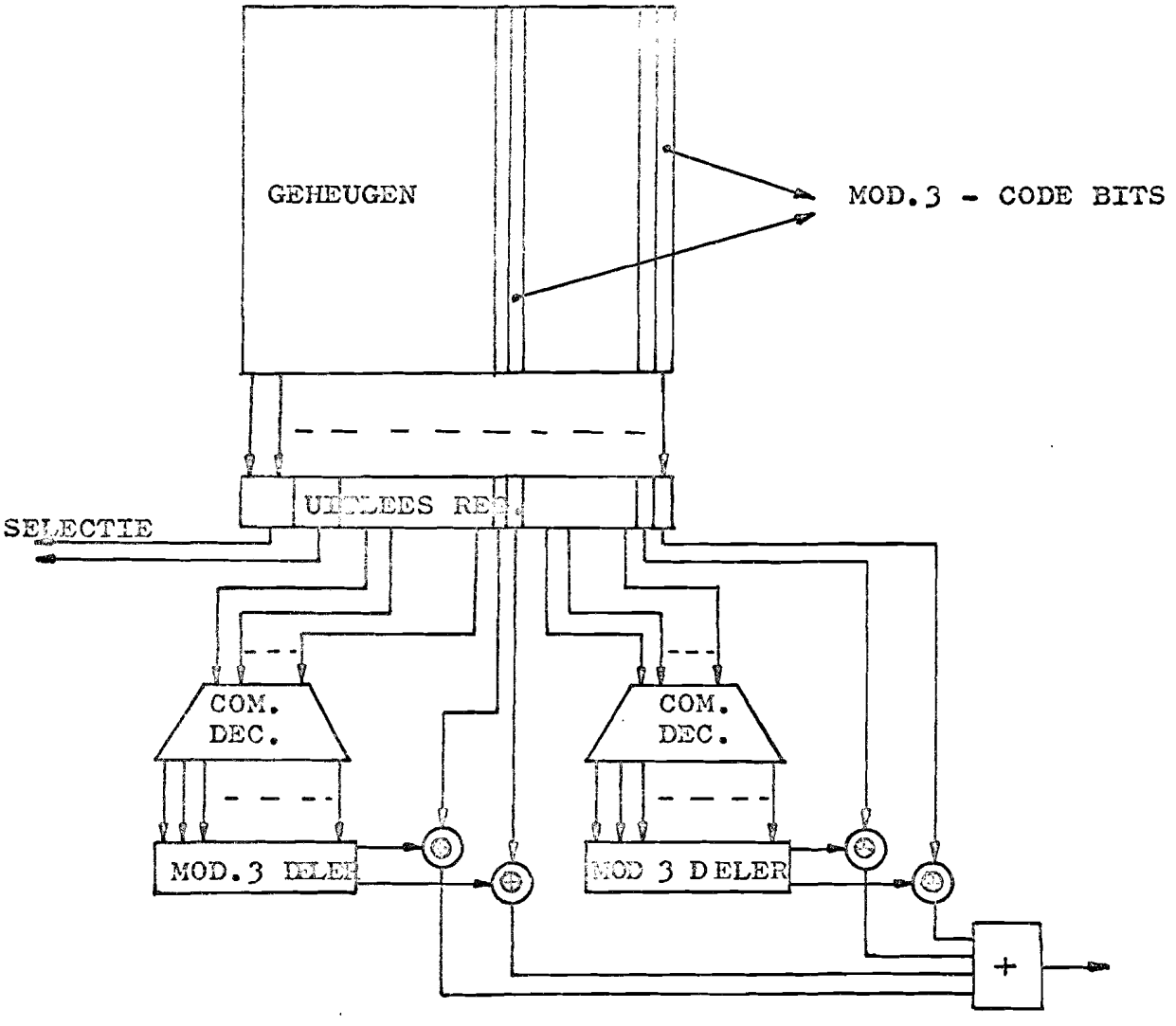

GEHEUGEN
MOD.3 - CODE BITS
SELECTIE
COM.
DEC.
COM.
DEC.
MOD.3 DELER
MOD 3 DELER
+

FIG. 20

5.3.3. Terug codering (fig. 19)

Decodering, zoals dit bij de commando-decodatie gebeurt, is een bepaalde code conversie, waarbij geen informatie verloren gaat. Ter contrôle op deze conversie kunnen we de uitgangsinformatie weer terug coderen naar de toestand zoals ze in oorspronkelijke vorm in het uitleesregister gegeven was. Door een simpel vergelijk kunnen we nagaan of bij de conversies fouten gemaakt zijn. Voor het vergelijk maken we gebruik van de reeds diverse malen genoemde methode met exclusive-or's.
De enige moeilijkheid die bij deze methode kan ontstaan is, wanneer meerdere mogelijke toestanden van het uitleesregister geen selectie van een commandolijn tot gevolg hebben, dan worden deze toch alleen op dezelfde wijze terug gecodeerd. Dit kan dan een niet-correcte foutsignalering tot gevolg hebben. Dit is te vermijden door alle toestanden, die binnen een veld geen selectie van een commandolijn tot gevolg hebben op dezelfde manier te coderen.

Beoordeling.

Deze methode beveiligt de decodatie door detektie van fouten. De kwaliteit is gelijk aan die van methode 5.3.1. Ook hier vindt detectie van alle fouten plaats.

De betrouwbaarheid is goed. Ook hiervoor gelden dezelfde opmerkingen als bij methode 5.3.1.

De detectiesnelheid is iets kleiner dan bij methode 5.3.1. De vertraging is 5 T . De extra vertraging is het gevolg van de looptijd van de terug-coderende poorten.

De kosten worden bij deze beveiliging ook weer uitsluitend bepaald door de benodigde logica. Ze bestaan uit een exclusive-or en een poort voor elk bit van het uitleesregister dat via de commando-decodatie uitgecodeerd wordt. Bovendien een or-schakeling voor het vormen van de foutsignalering. In totaal zijn hiervoor nodig 620 e.

Bij het schrijven van de microprogramma's moet rekening gehouden worden met de beveiliging. We moeten zorgen dat indien in een bepaald veld geen commandolijn geselecteerd wordt dit op eenduidige wijze gecodeerd wordt.

Levering als optie is ook hier zeer goed mogelijk.

5.3.4. Module 3 codering (fig. 20).

Bij het beveiligen van de commando-decodatie is het ook mogelijk gebruik te maken van het geheugen. Bij ieder veld in het geheugen nemen we een codewoord op, dat na decodatie van de veld informatie een controle op de juistheid van de decodatie mogelijk maakt.
Het codewoord zou in het eenvoudigste geval de pariteit kunnen zijn. Daar in alle gevallen dat een commandolijn geselecteerd wordt de pariteit van het gedecodeerde veld "1" is, heeft het weinig zin deze "1" in het geheugen op te nemen.
Beveiliging 5.3.b is reeds op deze pariteiteigenschap gebaseerd.
Ter beveiliging van de uitleesprocedure werden in hoofdstuk 5.2. een aantal pariteitbits voorgesteld. De mogelijkheid om deze bits bij de beveiliging van de commando-decodatie te gebruiken zou, uit systeemoverweging, een gunstige oplossing kunnen zijn.
Een code die de eigenschap heeft ook na conversie in een ander getallenstelsel haar eigenschappen te behouden is de modulo m code (2)(16)(17). Als codewoord wordt de rest bij deling van het informatiewoord door m gebruikt. Deze rest is altijd kleiner dan m. Als het informatiewoord vertaald wordt in een andere code, dan blijft de rest bij deling door m gelijk. Met andere woorden, hetzelfde codewoord ter detectie van fouten is zowel voor als na decodatie te gebruiken.
Als toepassing hiervan in de R.O.S. kunnen we bij ieder veld in het geheugen de rest bij deling door drie opnemen. Deze neemt per veld twee bits geheugenruimte in. Na decodatie van het veld delen we het gedecodeerde woord, dat gevormd wordt

door de commandolijnen en in een 1 uit n code staat, door drie en vergelijk-en het verkregen resultaat met de restcode in het uitleesregister. Dit geeft dan een contrôle op zowel het uitlezen als het decoderen. Door het toepassen van deze methode is het niet meer nodig de informatie in het uitleesregister met behulp van de toegevoegde code te controleren, daar dit gezamelijk met de controle op de commando-decodatie gebeurt. In het voorgaande hebben we voor ieder veld een restcode toegevoegd. Dit is niet noodzakelijk. We kunnen ook meerdere veld tegelijk met één codewoord beveiligen.

Beoordeling.

De beveiliging met een mod. 3 code detecteert alle enkelvoudige fouten en 50% van de tweevoudige fouten.
De beveiliging is zodanig dat ze tevens de woordinformatie beveiligt, waardoor de beveiligingslogica bij het uitleesregister overbodig wordt.

De detectiesnelheid is afhankelijk van de snelheid van de mod. 3 deler. Wordt hiervoor een seriedeler gebruikt, dan is deze langzaam. Een parallel deler is veel sneller. De snelheid hiervan wordt mede bepaald door het aantal bits van het te delen woord.
Bestaat het woord uit bv. 64 bits (4 velden), dan is de vertragingstijd ongeveer 9 τ . De kosten zijn bij deze beveiliging weer tweeledig. Enerzijds moet er geheugenruimte gereserveerd worden voor de code bits, en anderzijds is er een hoeveelheid logica nodig ten behoeve van de modulo 3 deler en de rest controle. Indien we een modub 3 deler gebruiken per 4 velden, dan hebben we voor de gehele beveiliging ongeveer 12 geheugenbits en 2500e nodig. De totale kosten zijn dus 4500e.

De beveiliging is niet als optie uit te voeren vanwege de grote invloed op het geheugen gebruik.

Opmerking. De modulo 3 test in het tweetallig stelsel komt overeen met de 11 proef in het tientallig stelsel.
De rest na deling door drie mag verkregen worden gedacht door het aantal enen van de enen, respectievelijk oneven bits modulo 3 op te tellen en vervolgens beide deelsommen modulo 3 samen te nemen.

5.3.5. Nabeschouwing over de beveiliging van de commando-decodatie.

De beveiligingsmethoden 5.3.1. decodatieverdubbeling en 5.3.3. terugcodering hebben ongeveer gelijke detectieeigenschappen. Beide beveiligingen zijn in staat om vrijwel alle typen fouten te herkennen en te detecteren. Uit de opbouw blijkt dat terugcoderen iets langzamer werkt. Bij het toepassen van terugcoderen als beveiliging moet rekening gehouden worden dat het niet selecteren van een lijn op eenduidige wijze wordt aangegeven. Voor bestaande systemen is terugcoderen daardoor niet altijd toepasbaar.

Pariteitbeveiliging 5.3.2. staat de detectie van enkelvoudige fouten binnen één veld toe. Ze is relatief duur. Aan logica kost ze ongeveer het dubbele van terugcoderen, terwijl de detectie mogelijkheden kleiner zijn.

Modulo 3 codering beveiligt zowel de commando-decodatie als de uitlees procedure. Qua kosten is deze beveiliging niet zo gunstig, doch uit systeemoverwegingen kan ze toch wel aantrekkelijk zijn.

5 4. Beveiliging van de adreskeuze.

Het geheugenwoord bevat naast informatie bestemd om commandolijnen te sturen ook gegevens betreffende het volgende te kiezen adres. Na uitlezen van het geheugenwoord verschijnen deze gegevens in het uitleesregister.
In bepaalde gevallen zijn ook toestanden buiten de R.O.S. van belang bij de keuze van een volgend adres. Daarom bestaat een adres uit twee gedeelten. Het ene, niet-conditionele, gedeel-

te wordt alleen door het geheugenwoord bepaald. Het kan daarom rechtstreeks van het uitleesregister naar het selectieregister worden overgebracht. Het andere, conditionele gedeelte, gaat naar het conditieregister. Hier zullen afhankelijk van de toestanden buiten de R.O.S. de conditionele bits gekozen worden. Ook deze worden in het selectie register geplaatst.

Bovenstaande informatiebewerkingen bestaan allen uit informatie transporten. Als beveiliging hiervoor zijn codeermethoden bijzonder geschikt. De keuze van de toe te passen code is afhankelijk van de gewenste detectiekwaliteit en de met de codering samenhangende kostenfactor. Pariteitbeveiliging zal vanwege de lage kosten en vaak voldoende detectiekwaliteit, hier zeer goed toegepast kunnen worden. Doch ook het toepassen van Hamming en andere codes kan overwogen worden.
Wanneer ter beveiliging van de uitleesprocedure een of meerdere bits gebruikt worden ter detectie van fouten in het nieuw te selecteren adres, dan kan dit adres direct in het selectieregister geplaatst worden. Foutdetectie hoeft dan alleen hier plaats te vinden.

Beoordeling.

De beveiliging kan afhankelijk van de toegepaste code enkel- of meervoudige fouten detekteren.

De detectiesnelheid wordt door de gebruikte decodeermethode bepaald. Ook hier zal parallel decodering aanmerkelijk sneller zijn dan seriedetectie. De detectiesnelheid wordt bovendien door de gebruikte code beïnvloed. Voor enkelvoudige pariteit controle zal de vertragingstijd ongeveer 2 τ zijn. De kosten zijn afhankelijk van de gebruikte code en codeermethode. Door het rechtstreeks overnemen van het codewoord uit het geheugen, worden de kosten ten behoeve van de coderingslogica uitgespaard. Daar het bij deze beveiliging slechts om een relatief klein aantal bits gaat, zijn de beveiligingskosten hiervoor niet groot. Voor codering en decodering met een pariteitbits kan volstaan worden met twee pariteitschakelingen met elk

10 à 16 ingangen, afhankelijk van de lengte van het adreswoord. De hiermee samengaande kosten zijn dan ongeveer 100e.

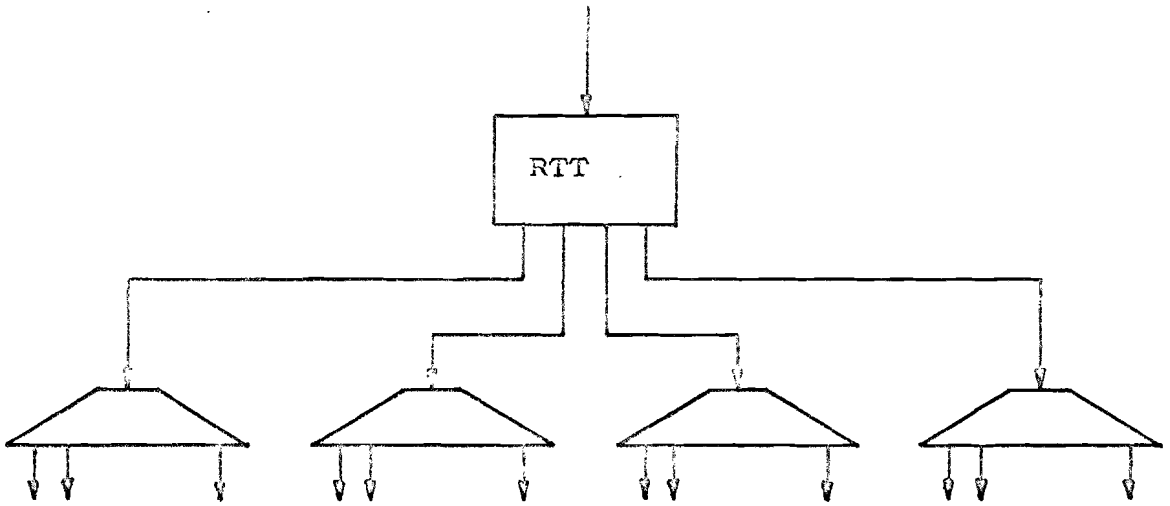
De beveiliging kan ook als optie uitgevoerd worden indien zowel codering als decodering plaats vindt.

5.5. Beveiliging van de register-tijden-teller (fig.21).

De register-tijden-teller ontvangt pulsen van de centrale klok. Hierdoor wordt de R.O.S. gesynchroniseerd met de rest van de rekenmachine.
De R.T.T. heeft vier uitgangen die elk om beurten een puls afgeven. Deze pulsen synchroniseren de interne informatiestroom van de R.O.S. De uitgangen van de R.T.T. moeten daartoe elk een groot aantal poorten sturen. De fan-out van een R.T.T. uitgang is daarvoor niet voldoende. Deze fan-out wordt verhoogd door achter de uitgang een aantal poorten te schakelen. Deze poorten kunnen elk weer een aantal andere poorten sturen. We noemen dit opbomen.
Wanneer we een R.T.T. hebben met een fan-out van 10, en tevens poorten met een fan-out van 10, dan kunnen we door 10 poorten te gebruiken de fan-out verhogen tot honderd. Stel dat we ook 100 andere poorten moeten sturen, dan is de fan-out in dit geval voldoende.
Willen we nu de R.T.T. beveiligen, dan kunnen we dit doen door verdubbeling. We plaatsen twee R.T.T.'s parallel aan elkaar. De uitgangen van beide vergelijken we met exclusive-or's. Indien een van beide R.T.T.'s faalt, dan wordt dit door deze exclusive-or's gesignaleerd.
Ter verhoging van de fan-out kunnen we nu elke R.T.T.-uitgang met 5 poorten verbinden. De totale fan-out is in dit geval ook 100 per uitgang.

Beoordeling.

De beveiliging heeft door verdubbeling goede foutdetecterende eigenschappen. Indien niet beide R.T.T.'s gelijktijdig dezelfde defecten vertonen, dan wordt iedere fout gedetecteerd.



NIET REDUNDANT

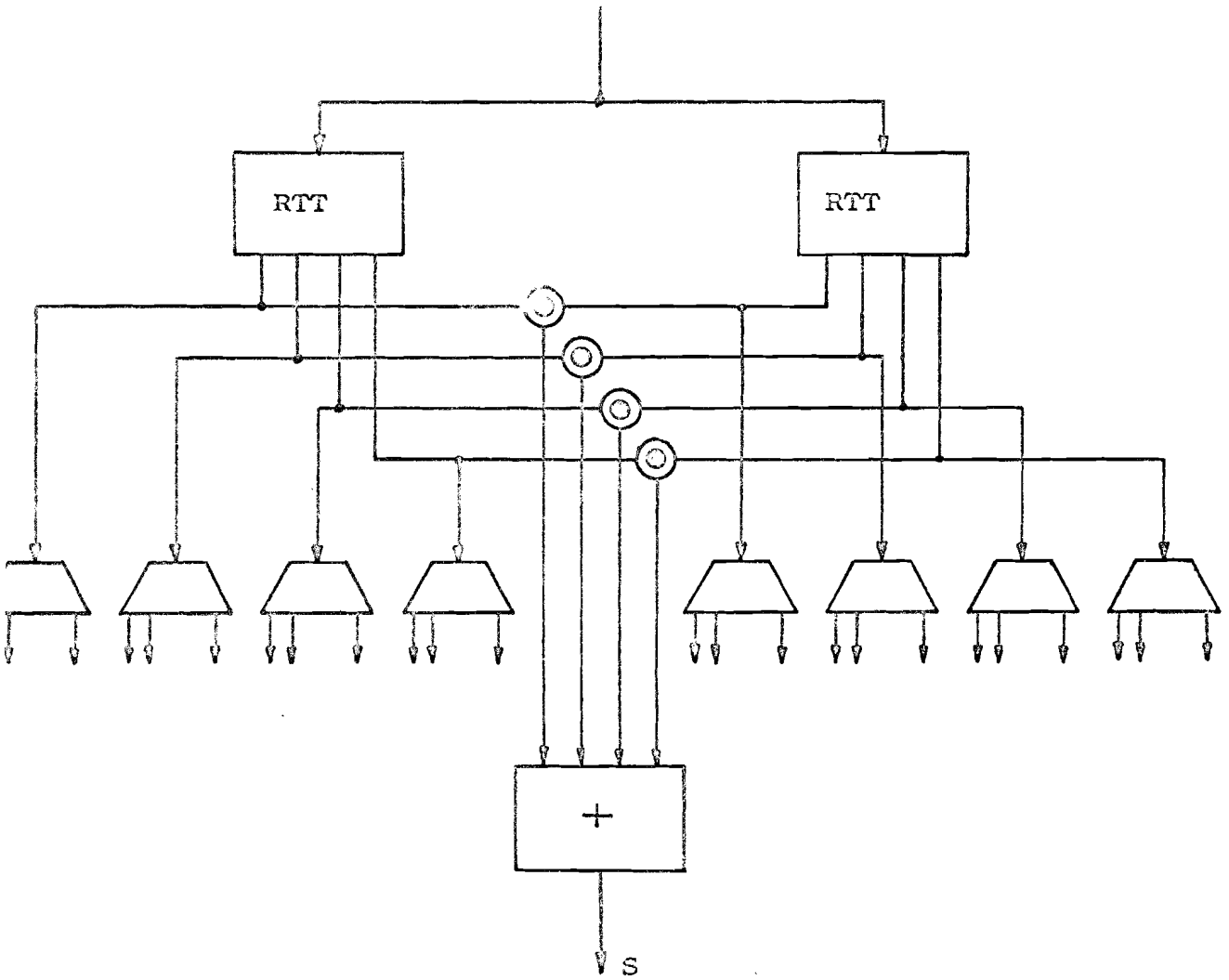


FIG. 21

De detectie geschiedt zeer snel. Alleen de vertraging van de exclusive-or's, ongeveer 2 τ , zijn hierop van invloed.

De kosten, in de vorm van logica, zijn een extra R.T.T. Stel dat deze 10e kost. De niet beveiligde R.T.T. kost dan inclusief de verhoogde fan-out schakeling 10+4x10e=50e. De beveiligde R.T.T. kost 2x10+4x10e=60e. De foutsignalering kost in dit geval dus 20% meer logica.

Een consequentie voor de ontwerper kan zijn dat uit veiligheidsoverwegingen gescheiden bomen ook op gescheiden plaatsen gebruikt moeten worden. M.a.w. het zal niet toegestaan zijn eenzelfde flipflop in te lezen met een stel commando's welke gesynchroniseerd zijn met verschillende R.T.T.'s, ook al lopen deze laatsten synchroon.
Zou dit wel gebeuren dan kunnen door pulsen, die synchroon zouden moeten zijn, t.g.v. verschillende propagatietijden van de R.T.T.'s met bijbehorende logica, flip-flops verkeerd gezet worden. De methode eist dus terugkoppeling naar de gebruiker van R.T.T. pulsen.

Constructie als optie is ook hier zeer eenvoudig mogelijk.

5.5. Overzicht van de beveiligingsmogelijkheden.

Als besluit van dit hoofdstuk over voorstellen voor de beveiliging van de diverse delen van de R.O.S., geven we een overzicht van de gedane beveiligingsvoorstellen.
In dit overzicht zijn de voorstellen op een aantal aspecten beoordeeld (fig. 22).
Hiermee is getracht een indruk te geven van de omvang van de kosten, uitgaande van de veronderstellingen betreffende afmetingen, zoals deze in dit hoofdstuk gehanteerd zijn. Om een iets duidelijker inzicht te krijgen is de verdeling van de kosten in andere situaties, zijn de kosten tevens gescheiden weergeven voor kosten aan logica (uitgedrukt in eenheden e) en kosten aan geheugenbits (uitgedrukt in aantal per woord).

De vertragingstijd van de beveiliging is uitgedrukt in aantallen maal de vertragingstijd van een poort t.

De aanwezigheid van de mogelijkheid een bepaalde beveiliging als optie te leveren is aangegeven met een + teken. De afwezigheid hiervan is met een - teken beoordeeld.

Voor de beoordeling van de kwaliteit van de beveiliging bleek het onmogelijk om in vergelijkbare cijfers, voor ieder geval aan te geven welk type fouten wel, en welk type niet gedetecteerd wordt. Daarom is een beoordelingsschaal ingevoerd van 0 tot 10. De cijfers die hierbij gebruikt zijn hebben de volgende betekenis.

10. Uitmuntend (alle fouten worden gedetecteerd)
9. Zeer goed (vrijwel alle fouten worden gedetecteerd)
8. Goed (een groot aantal type fouten worden gedetecteerd)
7. Ruim voldoende (meervoudige fouten worden gedetecteerd)
6. Voldoende (alle enkelvoudige fouten worden gedetecteerd)
5. Twijfelachtig (meeste enkelvoudige fouten worden gedetecteerd)
4. Onvoldoende (een klein aantal enkelvoudige fouten worden gedetecteerd)
3. Zeer onvoldoende (enkele fouten worden gedetecteerd)
2. Slecht (slechts een enkel bit beveiligd)
1. Zeer slecht (detectie van maximaal één bepaalde fout)

Bij deze beoordeling is geen exactheid nagestreefd, doch wel het aangeven van de richting waarin een mogelijke oplossing gevonden kan worden, uitgaande van vergelijkbare kosten aspecten etc.

BEOORDELING \ OPLOSSINGEN		5.1.1 verdubb.sel.dec.	5.1.2 toevoegen hulp dec.	5.1.3 adres verdubb.	5.1.4 adres codering	5.2. uitloos procedure	5.3.1 verdubb.com.dec.	5.3.2 pariteit contr.	5.3.3 terug decodatie	5.3.4 mod.3 codering	5.4. adreskeuze (pariteit)	5.5. R.T.T.
totale kosten ($\times 10^3$)		107	85	2,6	0,4	1,1	2,5	1,25	0,62	4,5	0,1	0,01
KOSTEN	geheugenbits			15	2	4				12		
	logica(e)	107.10^3	85.10^3	77	60	400	2500	1250	620	2500	100	10
vertraging beveiliging (t)		5	5	3	3	4	4	3	5	9	2	1
optie		+	+	-	-	-	+	+	+	-	+	+
KWALITEIT	selectie	9	8	8	6							
	uitlezen			3	2	7				8		
	commando decodatie						9	6	9	7		
	adres keuze										6	
	R.T.T.											7

Fig. 22

6. Toepassing van de beveiligingsmogelijkheden op een concrete Read Only Storage.

In het vorige hoofdstuk zijn diverse mogelijkheden genoemd die delen van de R.O.S. tegen het maken van fouten kunnen beveiligen. Voor elk van die mogelijkheden zijn tevens een aantal aspecten, die karakteristiek voor de beveiliging zijn zeer algemeen beoordeeld.
Om een voorstel ter beveiliging van een ROS te doen zullen we uitgaande van de genoemde mogelijkheden een oplossing moeten zoeken. Onontbeerlijk hierbij zijn concrete gegevens over de grootte, de foutkansen, cyclustijd etc. van de te beveiligen ROS. Ook zeer belangrijk zijn de foutkansen van de vertragingstijden van de toe te passen onderdelen.
Daar we over dergelijke gegevens in onvoldoende mate beschikken, zullen we slechts aantonen hoe de beveiliging van een R.O.S. tot stand zou kunnen komen. We zullen daarom eerst een aantal gegevens, eisen en veronderstellingen opsommen zoals deze ten dele uit praktijk gegevens beschikbaar zijn, en ten dele berusten op veronderstellingen die niet getoetst zijn. Verder zullen we aangeven hoe op grond van statistische beschouwingen een beveiligingsvoorstel tot stand kan komen.

6.1. Gegevens, eisen en veronderstellingen betreffende de Read Only Storage.

- De cyclustijd is 150 nsec.
- Het geheugen bevat 20000 woorden.
- Een woord bevat 120 bits.
- Het selectieadres wordt door 15 bits aangegeven, waarvan 3 bits conditioneel bepaald kunnen zijn.
- De commandodecodatie verzorgt de besturing van 300 commandolijnen.
- Een commandoveld bestaat uit 3 of 4 bits.
- De kosten drukken we uit in eenheden e.
- Een nand, and, or, nor, of een flipflop kosten elk één eenheid.
- De kosten van een exclusive-or zijn 5e.
- De kosten van een pariteitschakeling met n ingangen zijn 3ne.
- Het gebruik van geheugenbits is gunstig t.o.v. het gebruik van extra logica.

- Toegankelijkheid van de uitgecodeerde adresselectie is moeilijk te realiseren en dus gunstig.
- De failure-rate van een poort, flip-flop of leesversterker is 250 FIT *).
- Het aantal stoorpulsen in de uitleesversterker per bit is in dezelfde grootte orde als de failure-rate van poorten etc.

6.2. De Selectie.

Ten aanzien van de selectie merken we op dat oplossingen selectiedecodatieverdubbeling 5.1.1. en toevoegen van een hulpdecodatoe 5.1.2. minder geschikt zijn als beveiliging. Beide oplossingen vereisen toegankelijkheid van de uitgecodeerde adresselectie. Bovendien zijn ze kostbaar. Daarom zullen we deze methoden niet verder in onze beschouwingen betrekken.

Verdere mogelijkheden zijn adresverdubbeling 5.1.3. en adrescodering 5.1.4. Daar beiden principieel weinig verschillen, zullen we een voorstel doen afhankelijk van de, voor de selectiebeveiliging vereiste kwaliteit.
De benodigde kwaliteit wordt mede bepaald door de constructie van de selectiedecodatie. Bij een magnetische selectie is bijvoorbeeld de kans op selectie van meerdere adressen gelijktijdig groter dan bij een solid state selectie.
Door bepaalde fouten is het mogelijk dat met een magnetische selectie acht of meer adressen gelijktijdig worden gelezen.
Met een solid state selectie is een dergelijke fout zeer onwaarschijnlijk.

*) De failure-rate is gedefinieerd als het aantal defecten per onderdeel en per uur. Voor een nieuwe machine is dit aantal vrij hoog. Na de aanlooptijd blijft dit aantal ongeveer constant. In onze beschouwingen beperken we ons tot dit tijdsinterval. De daaropvolgende tijd, waarin verouderingsprocessen een rol spelen, ligt tegenwoordig ver na het verstrijken van de economische levensduur.
1 FIT is gedefinieerd als 10^{-9} fout per uur.

We zullen afleiden hoe goed de kwaliteit van de beveiliging voor een solid-state selectie moet zijn. We gaan daarbij uit van een selectie die is opgebouwd uit 10^4 poorten. Al deze poorten zijn volledig onafhankelijk van elkaar. Een defect van een poort heeft ook een enkelvoudige fout in de adresselectie tot gevolg. Verder gaan we uit van de veronderstellingen gemaakt in 6.1.
Het gemiddeld aantal fouten in de selectiedecodatie per geheugencyclus is dan

$$F = \frac{n.fr}{c} \approx 10^{-13} \text{ f/cyclus}$$

waarin: F = gemiddeld aan fouten per cyclus
n = aantal poorten
fr= failure rate van de poorten
c = aantal cycli per uur

De kans op een enkelvoudige fout gedurende één cyclus is volgens Poison

$$P(1) = e^{-F} \frac{F^1}{1!} \approx 10^{-13}$$

De kans op een tweevoudige fout is

$$P(2) = e^{-F} \frac{F^2}{2!} \approx 5.10^{-27}$$

Met deze gegevens over de foutkansen kunnen we het gemiddeld aantal fouten per jaar bepalen. Dit gemiddeld aantal verkrijgen we door het aantal cycli per jaar te vermenigvuldigen met de foutkans.
Het aantal enkelvoudige fouten per jaar is 20.
Het aantal tweevoudige fouten per jaar is 10^{-12}.
Dit laatste aantal is zo klein dat het overbodig is een beveiliging in te voeren die ook detectie van tweevoudige fouten geeft. Meervoudige fouten zijn nog onwaarschijnlijker, zodat we voor de beveiliging slechts rekening hoeven te houden met de enkelvoudige fouten. Een enkel pariteitbit is hiertegen een voldoende beveiliging.

Bij deze conclusie werd verondersteld dat één defect de selectie van een adres tot gevolg had, dat één bit afweek van het oorspronkelijke adres.

Kan één defect de selectie van het oorspronkelijke adres plus een adres dat één bit hiervan afwijkt tot gevolg hebben, dan kan detectie hiervan verkregen worden door bij het pariteitbit tevens het inverse bit hiervan op te nemen (5.1.4.).

Is bij magnetische selectie een grotere kans aanwezig op fouten waarbij meerdere adressen gelijktijdig uitgelezen worden, dan kan een beveiliging hiertegen verkregen worden door meerdere codebits plus hun inverse bits aan te brengen.
Zo kan men bijvoorbeeld het conditionele en het absolute adresgedeelte van een apart pariteitbit voorzien.

6.3. Het uitlezen.

Bij het uitlezen kunnen fouten ontstaan ten gevolge van een te kleine leesstroom, stoorpulsen, defecte leesversterkers, defecte register flip-flops en defecte verbindingen.
Een te kleine leesstroom wordt door de beveiligingsbits voor de adresselectie gedetecteerd. Daartoe kunnen de bij deze bits behorende, leesversterkers iets minder gevoelig gemaakt worden. Fouten als gevolg van een te kleine leesstroom zullen dan hier het beste merkbaar zijn.

We achten de kans op defecte verbindingen veel kleiner dan de kans op defecte flip-flops, leesversterkers en stoorpulsen.
De invloed hiervan zullen we daarom verwaarlozen.

Bij onze beschouwingen gaan we verder uit van de volgende gegevens:
Het aantal bit is n.
Het gemiddeld aantal stoorpulsen per bit per uur is $\mu_1 C$.
Het gemiddeld aantal defecte leesversterkers per bit per uur is $\mu_2 C$.
Het gemiddeld aantal defecte flip-flops per bit per uur is $\mu_3 C$.
Het aantal cycli per uur is C.

De kans dat een bit gedurende een clyclustijd foutloos blijft, is volgens Poison.

$$e^{-(\mu_1 + \mu_2 + \mu_3)}$$

De kans op een foutief bit is derhalve $p = 1 - e^{-(\mu_1+\mu_2+\mu_3)}$.
Daar ieder bit een gelijke kans op fouten heeft, en alle bits onafhankelijk van elkaar zijn, volgt met een binominale verdeling de kans op een enkelvoudige fout.

$$P(1) = \binom{n}{1} p\,(1-p)^{n-1}$$

Voor kleine pn kunnen we dit Poison benaderen, zodat

$$P(1) = e^{-pn} pn \simeq pn \simeq n\,(u_1 + u_2 + u_3)$$

Evenzo is de kans op een tweevoudige fout.

$$P(2) = \binom{n}{2} p^2\,(1-p)^{n-2}$$

Of met een Poison benadering

$$P(2) = e^{-pn}\,\frac{(pn)^2}{2!} \approx \frac{n^2}{2}\,(u_1 + u_2 + u_3)^2$$

Bepalen we hiervoor met onze gegevens de numerieke waarden, dan is:

$$P(1) = 1{,}25.10^{-15} \text{ en } P(2) = 0{,}8.10^{-30}$$

Het gemiddeld aantal enkelvoudige fouten per jaar is dan 0,27.

Het gemiddeld aantal tweevoudige fouten per jaar is dan $0{,}18.10^{-15}$.

Het aantal tweevoudige fouten is zo klein, dat beveiliging hiertegen overbodig is. Voor de beveiliging van de uitleesprocedure kan daarom volstaan worden met een enkel pariteitbit. Het is overbodig op te merken dat deze resultaten alleen valide zijn op voorwaarde dat de proposities, aan het begin van dit hoofdstuk, juist zijn. In praktische gevallen zullen deze waarden veelal verschillen. Bovenstaande conclusies hebben daarom zeker geen algemene strekking.

De beveiliging van de commando-decodatie kan op analoge wijze aan een beschouwing onderworpen worden. Daar dit op vrijwel identieke wijze gebeurt zullen we geen gedetailleerde beschrijving hiervan geven. We vermelden wel nog de resultaten. Het ge-

middeld aantal enkelvoudige fouten is 0,9 per jaar. Het gemiddeld aantal tweevoudige fouten is $0{,}38.10^{-15}$ per jaar. Verder zijn ook hier de voorgaande opmerkingen op van toepassing.

Ook bij de adreskeuze blijkt beveiliging tegen enkelvoudige fouten voldoende te zijn.

7. Slotbeschouwing.

Dit verslag heeft enkele belangrijke facetten die verband houden met het beveiligen van logische systemen, en in het bijzonder een Read Only Storage, bij elkaar willen brengen.
Enkele punten eruit zullen we nogmaals focusseren. Voor elk systeem geldt dat we moeten weten waartegen we willen beveiligen.
Willen we foutcorrectie, dan zijn hiervoor diverse methoden.
In het algemeen is dan slechts van een betrouwbaarheids verhoging sprake.
Willen we echter foutsignalering, dan zullen hiervoor detectiemethoden moeten worden gebruikt. Kostenaspecten zijn vaak de voornaamste redenen op grond waarvan een bepaalde beveiliging al dan niet zal worden toegepast. Hiermee zullen we dus voldoende rekening moeten houden. Ook de verdeling van de kosten over geheugenruimte en logica is zeer belangrijk.
Een methode die goede eigenschappen heeft om het uitlezen van meerdere adressen gelijktijdig te signaleren is het opnemen van de inverse bits bij de adrescode 5.1.4.
Uit systeemoverwegingen kan de modulo 3 codering gunstig zijn voor de beveiliging van het uitlezen en de commando-decodatie.
Statistische beschouwingen kunnen zeer belangrijk zijn bij het bepalen van de kwaliteit die een beveiliging moet hebben.
Bij de keuze en realisatie van een beveiliging zal altijd getracht moeten worden zoveel mogelijk gunstige aspecten voor het gehele systeem te verenigen. De beoordeling of aspecten al dan niet gunstig zijn is vaak zeer tijdgebonden. Door nieuwe technologische ontwikkelingen kunnen de onderlinge kostenverhoudingen van bijvoorbeeld logica, geheugenruimte en computertijd sterk veranderen. Ook de eisen die aan de kwaliteit gesteld worden zijn geenszins constant. De schrijver van dit verslag heeft dan ook niet de pretentie gehad een kant en klaar oplossing voor industriële toepassing af te leveren. Hij hoopt echter dat dit verslag van waarde zal zijn bij het bepalen van een beter gefundeerd standpunt betreffende beveiligingen.

YK
11.11.1968

Appendix I.

Bewijs behorende bij 4.1. (Verveelvoudiging van elementen)

We tonen aan dat $P_b > P_a > P_{syst}$. (zie 4.1 en fig. 6,7,8).

$P_{syst} = p^n$
$P_a = 1 - (1-p)^m$
$P_b = (1-(1-p)^m)^n$

We gaan uit van de volgende gegevens:

$0 < p < 1$; $q = 1 - p$; n, m geheel > 1

We moeten bewijzen dat I ; $P_a > P_{syst}$

II ; $P_b > P_a$

I $P_a > P_{syst}$, $1 - (1-p^n)^m > p^n$

Oplossing $0 < p < 1$ dus $0 < p^n < 1$ dus $0 < 1-p^n < 1$
dus $0 < (1-p^n)^{m-1} < 1$,
dan is: $(1 - p^n)^m < 1 - p^n$
en $1 - (1-p^n)^m > p^n$ QED

II $P_b > P_a$, $(1 - (1 - p)^m)^n > 1 - (1-p^n)^m$,
of $(1 - p^n)^m + (1 - (1 - p)^m)^n > 1$

Oplossing Zij $f(p) = (1-p^n)^m + (1 - (1 - p)^m)^n$
dan is $f'(p) = - nm\, p^{n-1}(1 - p^n)^{m-1} + mn(1-p)^{m-1}(1-(1-p)^m)^{n-1}$
Zij $g(p) = \dfrac{f'(p)}{nmp^{n-1}(1-p)^{m-1}}$

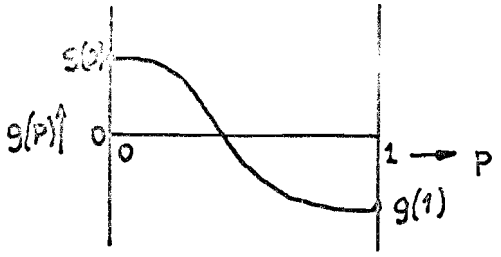
Dan is: $g(p) = \left(\dfrac{1-(1-p)^m}{p}\right)^{n-1} - \left(\dfrac{1 - p^n}{1 - p}\right)^{m-1} =$

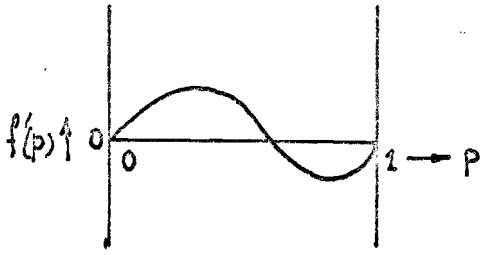
$= \left(\dfrac{1 - q^m}{1 - q}\right)^{n-1} - \left(\dfrac{1 - p^n}{1 - p}\right)^{m-1}$

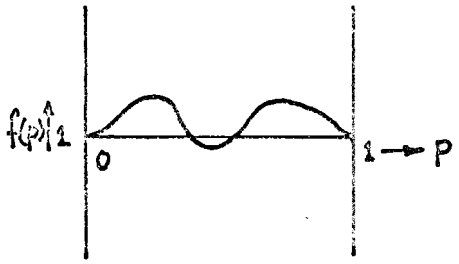
$= (1+q+ \ldots . q^{m-1})^{n-1} - (1+p+\ldots . p^{n-1})^{m-1}$

g is dalend in p, want de eerste term is stijgend in q, en dus dalend in p, en de tweede term is stijgend in p.

Verder: $g(0) = m^{n-1} - 1 > 0$, $g(1) = (1-n)^{m-1} < 0$.







$g(p)$ heeft dus precies één nulpunt voor $0 < p < 1$

Hetzelfde geldt dan ook voor $f'(p)$

Bovendien is $f'(p)$ positief, onmiddellijk rechts van $p = 0$ en negatief onmiddellijk links van $p = 1$.
Dat betekent dat $f(p)$ voor kleine positieve p groter dan 1, en evenzo voor kleine positieve q.

Als nu niet $f(p) > 1$ in $(0,1)$ geldt, dan is er minstens één punt waar $f(p)$ maximaal en minstens één waar $f(p)$ minimaal is. Dit kan niet omdat $f'(p)$ maar een nulpunt heeft.
Dus $f(p) > 1$ QED.

Appendix II.

Als voorbeeld bij 4.4 (Toepassen van codes ter beveiliging van logische systemen) wordt de beveiliging van een register fig. 23 besproken.

Allereerst zullen we dit register analyseren.
Uit A_i, B_i en P, Q , R worden de functies D_i en E_i gevormd.
$D_i = A_i\bar{B}_i\,P + A_i\,B_i\,Q\,R$

$E_i = \bar{A}_i\bar{B}_iP + \bar{A}_iB_iQ + A_i\bar{B}_iR + A_iB_i =$
$= \bar{A}_i\bar{B}_iP + B_iQ + A_iR + A_iB$

$\bar{E}_i = (A_i + B_i + \bar{P})(\bar{B}_i + \bar{Q})(\bar{A}_i + \bar{R})(\bar{A}_i + \bar{B}_i) =$
$= A_i\bar{B}_i\bar{R} + \bar{A}_iB_i\bar{Q} + \bar{P}\,\bar{A}_i\bar{B}_i + \bar{A}_i\,\bar{P}\,\bar{Q} + \bar{B}_i\bar{P}\bar{R}$

$D_i\bar{E}_i = A_i\bar{B}_i\,P\,\bar{R}$

We kiezen de waarden van de besturingslijnen P, Q, R zodanig dat $P\bar{R}$ altijd = 0 is.
Dan is dus ook $D_i\bar{E}_i = 0$.

Beschouwen we nu C_1 t/m C_4 dan is:

$\bar{C}_1 = \bar{E}_o + \bar{D}_o\bar{C}_o$ of $C_1 = E_oD_o + E_oC_o$

met $D_o\bar{E}_o = 0$ wordt $C_1 = D_o + E_oC_o$

Evenzo $C_2 = D_1 + E_1C_1$
$C_3 = D_2 + E_2C_2$
$C_4 = D_3 + E_3C_3$

Of $C_{i+1} = D_i + E_iC_i$

Verder is $S_i = D_iC_i + \bar{C}_{i+1}\,(E_i + C_i)$

$S_i = D_iC_i + (\bar{E}_i + \bar{D}_i\bar{C}_i)(E_i+C_i) = \bar{D}_iE_i \oplus C_i$

Met dit register zijn we in staat, afhankelijk van P, Q en R A_i en B_i te transporteren, op te tellen of af te trekken.

Voegen we aan de uitgangen S_i en C_4 een beveiligingscode toe, dan kan dit bijvoorbeeld een Hamming code zijn met de volgende voorschriften.

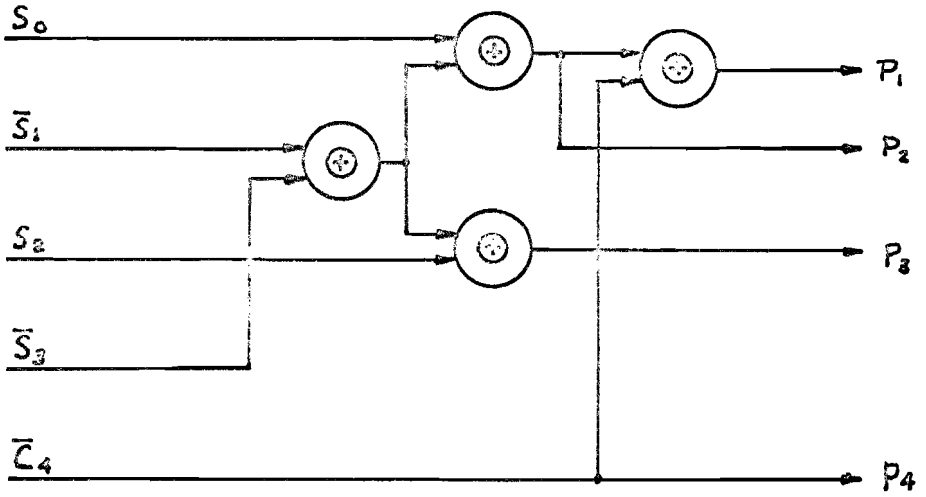
$P_1 = S_o \oplus \bar{S}_1 \oplus \bar{S}_3 \oplus \bar{C}_4$

$P_2 = S_o \oplus \bar{S}_1 \oplus \bar{S}_3$

$P_3 = \bar{S}_1 \oplus S_2 \oplus \bar{S}_3$

$P_4 = C_4$

Uitgaande van de uitgangen zijn P_1 - P_4 als onderstaand te construeren.



Opmerking. De looptijden zijn niet voor alle P gelijk.

Constructie van de code-bits

$P_1 = S_o \oplus \bar{S}_1 \oplus \bar{S}_3 \oplus \bar{C}_4 = C_o \oplus D_o \oplus \bar{D}_o E_o \oplus C_1 \oplus \bar{D}_1 E_1 \oplus C_3 \oplus \bar{D}_3 E_3 \oplus \bar{C}_4$
$= C_o \oplus C_1 \oplus C_3 \oplus \bar{C}_4 \oplus \bar{D}_o E_o \oplus \bar{D}_1 E_1 \oplus \bar{D}_3 E_3$

$P_2 = C_o \oplus C_1 \oplus C_3 \oplus \bar{D}_o E_o \oplus \bar{D}_1 E_1 \oplus \bar{D}_3 E_3$

$P_3 = C_1 \oplus \bar{C}_2 \oplus C_3 \oplus \bar{D}_1 E_1 \oplus \bar{D}_2 E_2 \oplus \bar{D}_3 E_3$

$P_4 = \bar{C}_4 = \bar{E}_3 \oplus \bar{D}_3 \bar{C}_3$

We zien dat een aantal $\bar{D}_i E_i$ termen modulo 2 opgeteld moeten worden. Daar deze termen onafhankelijk van elkaar zijn, kan er weinig vereenvoudigd worden.

Vereenvoudigingen kunnen worden aangebracht in de modulo 2 sommaties van de C_i's.

$C_1 \oplus C_3$ komt drie maal voor

$C_1 \oplus C_3 = (D_2 + E_2 C_2) \oplus C_1$

of uitgewerkt $C_1 \oplus C_3 = \bar{C}_1 D_2 + \bar{C}_1 E_2 D_2 + C_1 \bar{E}_2 + C_1 \bar{D}_2 \bar{E}_1$
verder is $\bar{C}_4 = \bar{E}_3 + \bar{D}_3 \bar{E}_2 + \bar{D}_3 \bar{D}_2 \bar{E}_1 + \bar{D}_3 \bar{D}_2 \bar{D}_1 \bar{C}_1$.

De constructie hiervan is weergegeven in fig. 24.

De oorspronkelijk benodigde hoeveelheid logica was 26 nands en 4 exclusive-or's. Voor de beveiliging zijn nodig 25 nands en 7 exclusive-or's.

We merken op dat dupliceren in dit geval eenvoudiger is. Ook door te trachten de realisatie op andere wijze uit te voeren kon geen verdere vereenvoudiging verkregen worden.

Wanneer we de uitgangen met een enkel pariteitbit wensen te beveiligen, dan kan dit op gelijke wijze gebeuren.

$P = S_o \oplus S_1 \oplus S_2 \oplus S_3 \oplus C_4$

$P = \bar{D}_o E_o \oplus \bar{D}_1 E_1 \oplus \bar{D}_2 E_2 \oplus \bar{D}_3 E_3 \oplus C_o \oplus C_1 \oplus C_2 \oplus C_3 \oplus C_4$

Voor de realisatie hiervan is het nodig dat weer alle E_i en D_i beschikbaar zijn. Verder moeten hier de modulo 2 som van alle C_i bepaald worden. De realisatie kosten hiervan zijn in dit geval ook niet veel lager dan de kosten van het register.

We kunnen hieruit concluderen dat, in dit geval, de kosten van een beveiliging met codes hoog, en voor een groot gedeelte constant zijn. Ze nemen slechts in geringe mater toe met de complexheid van de code.

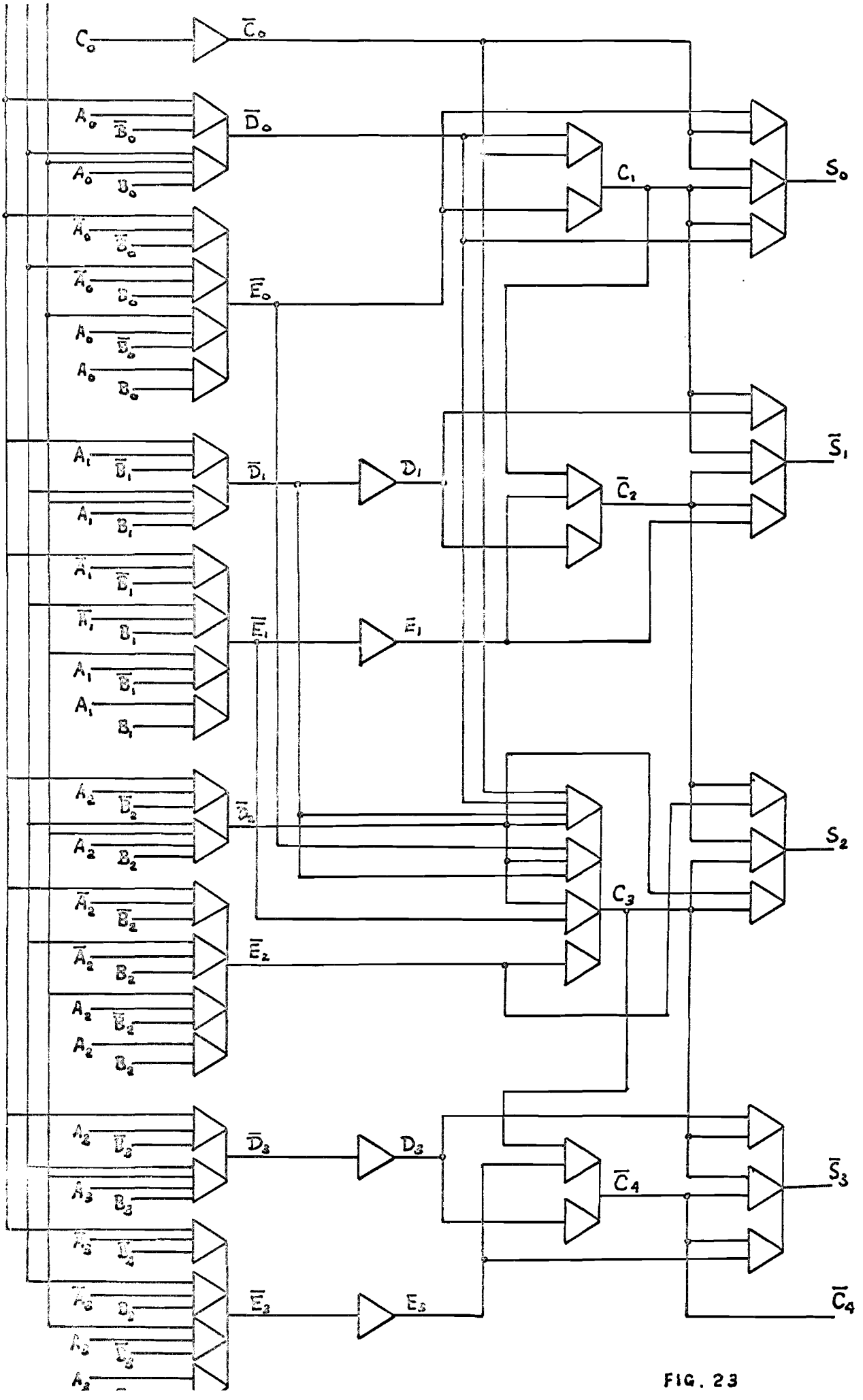

C0
C̄0
A0
B̄0
A0
B0
D̄0
Ā0
B̄0
Ā0
B0
A0
B̄0
A0
B0
Ē0
A1
B̄1
A1
B1
D̄1
D1
Ā1
B̄1
Ā1
B1
A1
B̄1
A1
B1
Ē1
E1
A2
B̄2
A2
B2
D̄2
Ā2
B̄2
Ā2
B2
A2
B̄2
A2
B2
Ē2
A2
B̄2
A2
B2
D̄3
D3
Ā3
B̄3
Ā3
B3
A3
B̄3
A3
B3
Ē3
E3
C1
C2
C3
C4
S0
S̄1
S2
S̄3
C̄4

FIG. 23

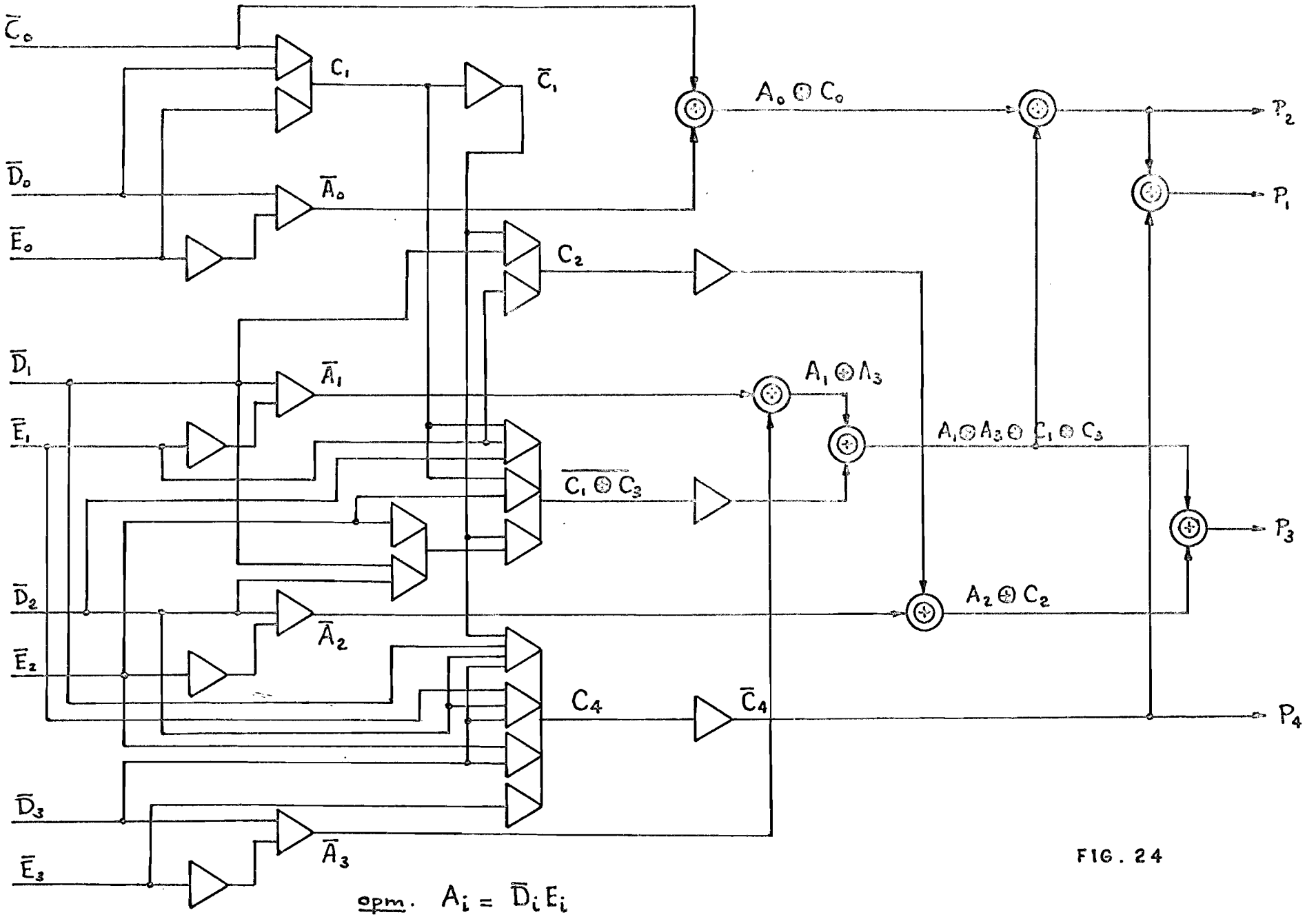


FIG. 24

opm. $A_i = \overline{D}_i E_i$

Appendix XII.

Enkele opmerkingen betreffende "Modulo 3-codering" (zie 5.3.4.).

Het uitleesregister is verdeeld in een aantal velden. Ieder veld bestuurt een aantal commandolijnen. De decodering van enkele velden beveiligen we met een modulo 3 code.

We gaan uit van bijvoorbeeld 3 velden a, b en c die elk respectievelijk l, m en n bits bevatten.

De bits noemen we a_o, a_1, ... a_{l-1}, b_o, b_1 ..., b_{m-1}, C_o, C_1, ... C_{n-1}.

Indien een willekeurig bit de waarde 0 heeft, dan is de rest bij deling door 3 ook gelijk aan 0. Heeft een bit de waarde 1, dan is de rest bij deling door 3 afhankelijk van de plaats van het bit. We zullen om dit te verduidelijken de verzameling velden nader analyseren.

Veld C bestaat uit n bits. Numeriek stelt het de volgende getalwaarde voor.

$$2^o.C_o + 2^1C_1 + 2^2.C_2 + ... + 2^{n-1}C_{n-1}$$

Bepalen we hiervan de rest bij deling door 3, dan is deze gelijk aan de rest bij deling door 3 van:

$$C_o + 2C_1 + C_2 + 2C_3 + ... \text{ mod. } 3(2^{n-1}).C_{n-1}$$

De rest bij deling door 3 verkrijgen we door de modulo 3 som te nemen van de bits met even index plus tweemaal de bits met oneven index.

Beschouwen we meerdere velden W, dan zijn deze als volgt te analyseren.

$$W = a\ 2^{m+n} + b\ 2^n + C, \text{ waarin:}$$
$$a = a_o + 2a_1 + 2^2a_2 + ... + 2^{l-1}.\ a_{l-1}$$
$$b = b_o + 2b_1 + 2^2b_2 + ... + 2^{m-1}\ b_{m-1}$$
$$c = C_o + 2C_1 + 2^2C_2 + ... + 2^{n-1}\ C_{n-1}$$

De rest bij deling door 3 is dan:

$$\begin{aligned}&\text{mod } 3\,(a).\ \text{mod } 3\,(2^{m+n}) + \text{mod } 3(b).\ \text{mod } 3(2^{n}) + \text{mod } 3(c)\\ &= C_o + 2C_1 + C_2 + \ldots. C_{n-1}.\ \text{mod } 3\,(2^{n-1}) + b_o \text{ mod } 3(2^{n}) +\\ &+ b_1.\ \text{mod } 3(2^{n+1}) + \ldots. + b_{m-1}.\ \text{mod } 3\,(2^{m+n-1}) +\\ &a_o.\ \text{mod } 3(2^{m+n}) + a_1 \text{ mod } 3\,(2^{m+n+1}) + \ldots + a_{l-1} \text{ mod } 3\\ &(2^{l+m+n})\end{aligned}$$

Beschouwen we alle bits in volgorde, en kennen we ze de indices toe als volgt:

$$C_o, C_1, C_2, \ldots C_{n-1}, b_{n+o}, b_{n+1}, b_{n+2}, \ldots b_{n+m-1}, a_{n+m+o}, \ldots a_{n+m+l-1}$$

dan verkrijgen we de rest bij deling door 3 door mod 3 sommatie van de bits met even index plus twee maal de bits met oneven index.

Na decodatie krijgen we bij de commandolijnen de volgende situatie.

Veld c wordt uitgecodeerd tot 2^n lijnen.
Deze noemen we $r_o, r_1, r_2 \ldots r_{n-1}$.
Veld b wordt uitgecodeerd tot 2^m lijnen.
Deze noemen we $9_o, 9_1, 9_2, \ldots 9_{2^m-1}$.
Veld a wordt uitgecodeerd tot 2^l lijnen.
Deze noemen we $p_o, p_1, p_2, \ldots p_{2^l-1}$.

De indices o, 1, ... geven daarbij steeds de waarde van de bijbehorende waarden c_i, b_i, a_i aan.

Noemen we de gedecodeerde waarde van W W^1, dan is deze numeriek gelijk aan W.
Uitwerking geeft:

$$W^1 = r_o + r_1 + \ldots r_{2^n-1} + (9_o + 9_1 + \ldots 9_{2^m-1})\ 2^n + (p_o + p_1 + \ldots + p_{2^l-1})\ 2^{n+m}.$$

De rest bij deling door 3 is gelijk aan de mod 3 som van

$$\begin{aligned}&0.r_o + 1r_1 + 2r_2 + 0r_3 + \ldots \text{mod } 3(2^n-1).r_{2^n-1} + (0.9_o + 1.9_1 + \ldots +\\ &+ 9_{2^m-1} \text{ mod } 3(2^{mn}-1)).\text{mod } 3\,(2^n) + (0.p_o + \ldots p_{2^l-1} \text{ mod } 3(2^l-1)).\\ &(\text{mod } 3(2^{n+m})).\end{aligned}$$

waarin $\text{mod } 3\,(2^p) = \begin{cases} 1 & p = \text{even} \\ 2 & p = \text{oneven} \end{cases}$

Literatuurlijst.

1. Wilcox & Mann:
"Redundancy techniques for computing systems".
Spartan books.

2. W.W.Peterson
"Error correcting codes".
M.I.T. press.

3. W.H. Pierce
"Failure tolerant computer design".
Academic press.

4. J. Swoboda
"Binäre Gruppencodes zur Sicherung logischer Schaltkreise gegen Fehler".
Elektronische Rechenanlagen 7 (1965) H.2, S.85-90.

5. F.Zendeh
"Selbstkorrigierende Zuordner für vollständige Codes".
Kybernetik 2 (1964) S.114-124.

6. A.G.Franco, N.Marchand, L.J.Sparta
"Error-control systems get the message across".
Electronics, 15 nov. 1965, pp.125-127 Vol.38 no 23.

7. H.D.Burton, E.J.Weldon
"Cyclic product codes".
I.E.E.E. trans. on I.T. july 1965, pp.433-439.

8. J.Hartmanis, R.E.Stearns
"A study of feedback and errors in sequential machines".
I.E.E.E. trans. on E.C. june 1963, pp.223-232.

9. E.Elspas, J.K.Wolf
"Error locating codes- a new concept in error control".
I.E.E.E. trans. on I.T. april 1963, pp. 113-117.

10. Ph.I.Hershberg
"A class of error-detecting codes with absent parity bits".
I.R.E. trans. on C.S. sept. 1962, pp. 280-284.

11. C.V. Freiman
"Optimal error detection for completely asymetric binary channels".
Information and control 5, 1962, pp. 64-71.

12. D.T.Brown, W.W.Peterson
"Cyclic codes for error detection".
Proc. of the I.R.E. jan. 1961, pp. 228-235.

13. D.B.Armstrong (Bell labs)
"Error detecting and correcting system",
U.S. Pat. 3.128,449 Issued 7, 1964.

14. R.W.Hamming
"Error detecting and error correcting codes".
The B.S.T.J. Vol. XXVI, april 1950, no 2, pp. 147-160.

15. H.Weiher
"Möglichkeiten und Grenzen der Zuverlässigkeitserhöhung durch Redundanz in der Nachrichtentechnik Teil 1".
Nachrichtentechnik 16 (1966) H.4. p. 152-157.

16. C.W.Hastings, R.W.Watson
"Self checked computations using residue arithmetic".
Proc. I.E.E.E. dec. 1966, vol. 54, no 12, pp. 1920-1931.

17. T.G.Gadess
"A study of an error detecting parallel adder".
U.S. Gov. Res. and Dev. Report AD 647, 152, jan. 1967.

18. J.B.Connolly, W.G.Schmidt
"Failure Erasure circuitry: a duplicate technique for failure-making systems".
U.S. Gov. Res. and Dev. Report, AD 477, 218 oct. 1965.

19. H.D.Goldman
"Protecting Core memory circuits with error correcting cyclic codes".
I.E.E.E. trans. on E.C. june 1964, pp. 303-304.

20. W.W.Peterson, M.O.Rabin
"On codes for checking logical operations".
I.B.M. journal, april 1959, pp. 163-168.

21. W.G.Brown, J.Tierney, R.Wasserman
"Improvement of electronic computer reliability through the use of redundancy".
I.R.E. trans. on E.C.-10, 1961, pp. 407-416.

22. S.E.Lomax
"Research on failure free systems" (Final report).
U.S. Gov.Res. and Dev.Report, N 66-27033, dec 1963.

23. J.L.Massey
"Survey of residue coding for arithmetic errors".
I.C.C. Bulletin, vol.3, pp.195-209, oct. 1964.

24. C.J. Greveling
"Increasing the reliability by the use of redundant circuits".
I.R.E. proc. 44 I, pp. 509-515.

25. I.P. Lipp
"Topology of switching elements vs. reliability".
I.R.E. trans. on rel. & quality contr. june 1957, pp.21-23.

26. M. Kochen
"Extension of Moore Shannon model for relay circuits".
I.B.M. journal, april 1959, pp.169-186.

27.
"Self repair techniques investigation".
U.S. Gov.Res. and Dev.Report, AD 647 831, 1967.