Inleiding tot het automatisch ontwerpen van digitale systemen

van Gastel, H.F.N.

Award date:
1975

AFDELING DER ELECTROTECHNIEK
TECHNISCHE HOGESCHOOL
EINDHOVEN

Groep ECB

INLEIDING TOT HET AUTOMATISCH ONTWERPEN VAN DIGITALE SYSTEMEN

door H.F.N. van Gastel

Rapport van het afstudeerwerk
uitgevoerd van 7.6.1971 tot 22.6.1972
in opdracht van Prof. Ir. A. Heetman
onder leiding van Ir. J. Hoogeveen

INHOUDSOPGAVE.

1. INLEIDING.

Aan de Technische Hogeschool Eindhoven is een systeem ontwikkeld voor het automatisch ontwerpen van digitale systemen. (zie lituratuuropgave 1) Het ontwerpen geschiedt door het systeemgedrag van het te ontwerpen systeem te beschrijven in de ontwerptaal DSL2 (Digital System Language 2) waarna door een aantal computerprogramma's de DSL2 beschrijving wordt omgezet in reele circuits.
De gedragsbeschrijving in DSL2 is geformaliseerd in de procedure. Uit deze gedragsbeschrijving worden de Boolse functies van het systeem afgeleid. Daarna worden de functies omgezet in circuits met hun verbindingen. Het ontwerpen van het systeem houdt dus de omzetting in van de procedures naar declaraties. Nadat het systeem ontworpen is kan het worden gesimuleerd en na de bouw worden getest.

Mijn afstudeeropdracht hield een lituratuurstudie in over het automatisch ontwerpen van digitale systemen en het [illegible] genereren van Mm extended partition paren.
Toelichting:
Bij de lituratuurstudie over het automatisch ontwerpen van digitale systemen lag vooral de nadruk op de vraag, in hoeverre met reeds bestaande talen automatisch kon worden ontworpen. Het genereren van de Mm extended partition paren was nodig omdat de Mm-EPP's alle informatie gaven voor het toewijzen van variabelen aan de toestanden van een sequentiele machine. Toen het programma voor het genereren van de Mm extended partition paren al in een vergevorderd stadium verkeerde, bleek de theorie uitgebreid te moeten worden tot MMm triplets met als gevolg dat het genereren van Mm paren nutteloos werd. De resterende tijd is daarom gevuld met het nader bestuderen van de theorie over het automatisch ontwerpen van digitale systemen beschreven in DSL2. Dit verslag kan dan ook gezien worden als een afspiegeling van deze studie.

Gaarne wil ik de Hooggeleerde Heer Prof. Ir. A. Heetman dank zeggen voor de zeer gewaardeerde hulp en raadgevingen welke ik gedurende mijn studie van hem mocht ontvangen.
Tevens wil ik de Weledelgestrenge Heer Ir. J. Hoogeveen bedanken voor diens hulp, welke een belangrijke bijdrage heeft geleverd aan het tot stand komen van mijn afstudeerwerk.

2. TALEN VOOR DE BESCHRIJVING VAN DIGITALE SYSTEMEN.

Veelal wordt een digitaal systeem beschreven door middel van Boolse vergelijkingen en logische diagrammen. Er zijn echter diverse redenen om een digitaal systeem te beschrijven door gebruik te maken van een formeel gedefinieerde taal:

1. Systeembeschrijving is eenduidig.
2. Met behulp van een ontwerptaal wordt de ontwerpdocumentatie en de communicatie tussen de ontwerpers eenvoudiger.
3. Door middel van een computerprogramma(translator) kan een formele systeembeschrijving worden omgezet in een aantal Boolse vergelijkingen. Deze vergelijkingen kunnen door een general-purpose simulator worden gesimuleerd. Door de toepassing van de computer vermindert de ontwerpduur.
4. De logische simulatie kan gebruikt worden om de uitvoering van het systeem te bepalen en de logische vergelijkingen om de kosten van het ontwerp te berekenen.
5. Met een computer-ontwerptaal en een translator kunnen de ontwerpprocessen van vertaling, simulatie, logische minimalisering, ontwerpverandering, kostenschatting, bepaling van de uitvoering, kaart-layout, bedrading en documentatie worden geautomatiseerd op een geïntegreerd niveau. Dit verkort de ontwerptijd en verlaagt de ontwerpkosten.
6. M.b.v. een taal kunnen complexere machines worden ontwikkeld.
7. Het gebruik van een taal veroorzaakt een vergroting van de creativiteit door de man-machine interactie.

Om een digitaal systeem met een computer te kunnen ontwerpen zal de taal, waarin het systeem wordt beschreven, aan een aantal eisen moeten voldoen:

1. De taal moet dicht bij de natuurlijke taal staan zodat deze geschikt is voor ontwerpdocumentatie. Bestaande systemen moeten gemakkelijk kunnen worden gedefinieerd.
2. Zij moet formeel gedefinieerd zijn.
3. Zij moet om te zetten zijn in Boolse vergelijkingen.
4. Zij moet middelen bevatten om parallelle bewerkingen, timingsignalen, control-commando's en serie- of parallel-transfers uit te drukken.

5. De taal moet onafhankelijk van iedere techniek zijn, d.w.z. open.
6. De systeembeschrijving moet vrijwel dezelfde organisatie hebben als het systeem zelf.
7. De taal moet eenvoudig en compact zijn zonder dat de vereenvoudiging leidt tot een krachteloze taal.

Door een bestaand systeem te beschrijven in een general-purpose taal kan het systeem worden gesimuleerd. (2)/(5). De beschrijving wordt vertaald door een bestaande compiler naar een programma, wat de simulatie uitvoert. Door de simulatie komen de 'bottleneck'punten in het systeem aan het licht en kan de hardware-uitvoering van het systeem worden bepaald. Ook zal het effect blijken van verschillende schema's die betrekking hebben op de software-uitvoering. Het nadeel van een simulatie is, dat deze zoveel tijd vergt.

Uit een systeembeschrijving kunnen ook logische vergelijkingen worden gegenereerd. Een dergelijk systeem kan bestaan uit een translator, een tijdanalyse-routine en een generator voor logische vergelijkingen. De input van het systeem bestaat uit 3 delen:

1. een structurele beschrijving van het systeem; registers, klok en informatie-flowwegen.
2. de elementaire instructies die moeten worden uitgevoerd alsmede de overgang naar de volgende instructie. De instructies mogen tijdbeperkingen bevatten.
3. een bibliotheek van logische bouwstenen zoals optellers, vermenigvuldigers, registers enz.

De translator heeft tot taak om uit input 1 en 2 een ontwerptabel te vormen welke bestaat uit een beschrijving van alle gebruikte registers, hun tijdkarakteristieken en de logische voorwaarden welke de transfer van informatie over de gedefinieerde informatiepaden regelt. De tijdroutine bepaalt wanneer iedere informatieoverdracht mag beginnen. De logische vergelijkingsgenerator combineert de ontwerptabel en de tijdinformatie en leidt via logische conjuncties en disjuncties de logische

vergelijkingen af.
Wilkes en Stringer (6) waren een van de eersten die met een ontwerptaal de specificatie, documentatie en het ontwerp van digitale systemen wilden vereenvoudigen. Helaas is de taal niet algemeen bruikbaar omdat ze teveel gericht is op de digitale machine die door Wilkes en Stringer werd ontwikkeld.

Reed's register transfer taal (7)/(9) is meer toegepast. De taal is gemakkelijk te leren, vrij algemeen toepasbaar en de statements associeren direct met hardware. Een volledige beschrijving van het systeem kan echter in deze taal niet worden gegeven, er bestaan geen voorzieningen voor deelsystemen en de kleine vocabulaire noodzaakt het gebruik van vele symbolen.
Het werk van Schorr (9) beschrijft zowel de omzetting van een systeembeschrijving in een registertransfer-taal naar logische vergelijkingen, als de omzetting door een compiler van logische vergelijkingen naar registertransfers.

De LDT taal van Burroughs (10)/(12) en de Sequence Chart van IBM (13) vullen de taal van Reed zodanig aan, dat een volledige beschrijving van het systeem kan worden gegeven. Het klassieke werk van Gorman en Anderson (10) voor het genereren van logische vergelijkingen uit de systeembeschrijving werd aangevuld door Proctor (11) en verder verfijnd door Estrin en Mandell(12). De Sequence Chart(13) vertoont timing en de opeenvolging van informatie op een grafische manier. Beide talen worden gesteund door computerprogramma's welke logische vergelijkingen genereren van Reed-achtige statements. De grafische aard van de Sequence Chart is echter onhandig voor het automatisch verwerken.

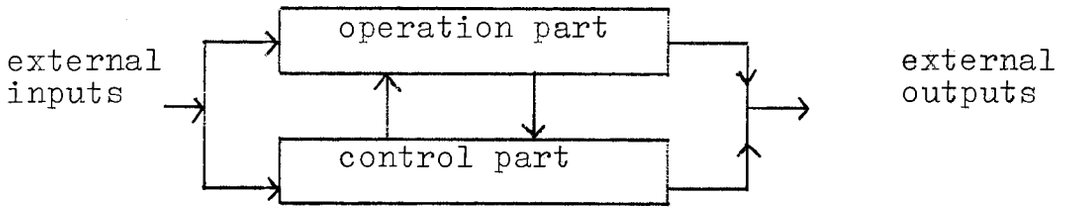
APL, de taal van Iverson (14)/(17), heeft een grote vocabulaire en is universeel bedoeld. De taal is geïmplementeerd als programmeertaal voor een timesharing-systeem en is machineonafhankelijk. APL biedt een erg beknopte en nauwkeurige notatie om transfers tussen registers te beschrijven. Een groot deel van de taal is echter niet te gebruiken bij automatisch ontwerpen, enkele symbolen zijn niet eenvoudig te associeren met hardware, terwijl timing en sequencing niet gemakkelijk beschreven worden.
Anderen hebben Algol-achtige ontwerptalen voorgesteld (18)/(21). Deze talen zijn minder dan voldoende: symbolen en syntax welke

gericht zijn op algorithmische berekeningen zijn meestal geen goed middel om hardware te beschrijven.
Het CASD systeem van IBM (22) maakt gebruik van een ontwerptaal die afgeleid is van PL/I en uitgebreid is met een aantal statements om parallelle bewerkingen uit te voeren en statements voor synchronisatieaspecten. Bij het CASD-systeem zijn de logische translatie en de logische simulatie in één systeem gecoördineerd. Het beschreven systeem in de source design language wordt door een computerprogramma (de encoder) omgezet in een internal form. Deze internal form is zowel de input voor een hoog-niveau simulator als voor een aantal translatoren die de beschrijving omzetten in logische specificaties. De timing is in het CASD-systeem niet zo expliciet vermeld als in de andere talen. Er is hier gebruik gemaakt van het asynchrone ontwerp zoals dat voorgesteld is door Metze en Seshu (23).
De taken worden gesynchroniseerd door het gebruik van verdeelde variabelen en WAIT statements. De ontwerptaal is opgebouwd uit procedures. Omdat de inleiding van literatuuropgave (22) vermeldt dat er geen plannen bestaan het werk op dit gebied voort te zetten, zou kunnen worden geconcludeerd dat het CASD-systeem niet aan de verwachtingen voldoet.

De laatste jaren is het onderzoek naar talen ter beschrijving van digitale systemen meer verplaatst naar de universiteiten. Zo ontwikkelde Duley (24) aan de universiteit van Wisconsin, Madison, een taal DDL (Digital system Design Language). De taal is onafhankelijk van iedere speciale technologie, ontwerpprocedure, machineorganisatie enz.; staat specificatie toe op ieder niveau van detail; is mnemonic en fundamenteel en zodoende zeer leesbaar en heeft een syntax en semantiek die een documentatie toestaat met een organisatie die parallel loopt aan de blokstructuur van het te beschrijven systeem. DDL kan worden getransformeerd in een ander DDL document bestaande uit Boolse vergelijkingen voor combinatorische logica en volgende toestandsvergelijkingen voor flipflops (25).

Een interessant werk is verricht door Giovanni B. Gerace (26) met medewerking van de wiskundige afdeling van de universiteit

van Cambridge.
Om een digitaal systeem te beschrijven gebruikt Gerace een register transfer taal welke aangepast is aan een bepaalde door hem ingevoerde systeemstructuur. In figuur 1.1 is deze structuur aangegeven.



figuur 1.1: algemene structuur van een digitaal systeem zoals gebruikt door G.Gerace (26)

De ontwerpmethode kan nu als volgt worden samengevat:
1. Het systeem wordt beschreven in een register transfer taal
2. De microprogramma's van de systeembeschrijving worden m.b.v. een algorithme vertaald in een aantal flowtables
3. De flowtables worden door een synthese omgezet in overeenkomstige sequentienetwerken.

Het resultaat van stap 2 maakt het mogelijk om de logische complexiteit van een systeem te reduceren door gebruik te maken van systhematische technieken van de sequentiele netwerktheorie. Het ontwerpen van digitale systemen op deze manier is echter beperkt tot LLC-circuits (Level-input, Level-output, Clocked-sequential circuits) welke iteratieve asynchrone netwerken mogen bevatten.

Aan de afdeling toegepaste wiskunde van de universiteit van Grenoble is de taal CASSANDRE ontwikkeld. CASSANDRE (27) is zowel een taal voor Computer Aided Design, Simulation of logical Systems their Analysis, Description and Realisation, als ook een systeem dat ontworpen is voor conversationele uitvoering en uitwerking van een machine beschreven in deze taal. De synthese van een beschrijving in CASSANDRE geschiedt door een verfijning van de initiele verdeling van de oorspronkelijke beschrijving via de syntax. Het doel is om te komen tot een beschrijving met alleen elementaire units en signalen tussen deze units. Hierbij wordt de ontwerpmethode van Gerace (26) toegepast.

Interessant is het feit, dat de taal CASSANDRE reeds gebruikt wordt door hardware studenten aan de universiteit van Grenoble om in een conversationeel proces de logische ontwerpen te simuleren. Dit bespaart diverse hardware uitvoeringen. De ontwerpkwaliteiten van CASSANDRE zullen door het gebruik van de ontwerpmethode van Gerace (26) beperkt zijn tot LLC circuits.

Om een digitaal systeem met een computer te ontwerpen zal de systeembeschrijving door een aantal computerprogramma's moeten worden omgewerkt tot een beschrijving die geschikt is voor de hardware uitvoering. In het voorgaande gedeelte is speciaal de omzetting van de systeembeschrijving in Boolse vergelijkingen beschouwd. Deze Boolse vergelijkingen zullen door programma's moeten worden omgewerkt tot het uiteindelijke ontwerp. Voor de problemen die hierbij een rol spelen zij verwezen naar het algemeen overzicht van het automatisch ontwerpen van digitale computers, geschreven door Breuer (28).

Uit het voorgaande is gebleken, dat het moeilijk is om een taal te definieren welke aan alle reeds besproken eisen voldoet.
Het beste voldeden de talen DDL en CASSANDRE welke speciaal ontwikkeld werd voor het ontwerpen van digitale systemen.
De besproken register transfer talen bleken niet voldoende om ieder digitaal systeem te beschrijven. De talen die afgeleid waren van hogere programmeertalen leverden geen geschikte ontwerptalen op.
Het is duidelijk dat tot nog toe geen enkele taal bestaat waarmee een digitaal systeem kan worden beschreven zodanig dat door de computer een aantal alternatieve systeemontwerpen kunnen worden gegenereerd die allen voldoen aan de initiele eisen en aan een of meerdere criteria. Er kan pas van <u>automatisch</u> ontwerpen van digitale systemen worden gesproken als uit een gedragsspecificatie automatisch een functioneel ontwerp kan worden geanalyseerd en gerealiseerd. Dit mag wel worden uitgevoerd in een conversationeel proces: de computer genereert een aantal oplossingen en de ontwerper kan er een kiezen. Bij het automatisch ontwerpen moet de computer in staat zijn om bouw-

stenen te genereren, die in de oorspronkelijke beschrijving niet voorkomen maar uit de synthese van de gedragsspecificatie wel nodig blijken te zijn.

Aan de Technische Hogeschool Eindhoven is de taal DSL2 (Digital System Language) ontwikkeld waarmee het mogelijk is om digitale systemen automatisch te ontwerpen. Met DSL2 wordt een systeem beschreven in procedures en declaraties. De procedures beschrijven de bewerkingen die door het systeem moeten worden uitgevoerd. De declaraties beschrijven de elementen van het systeem. Het digitale systeem is ontworpen als alle procedures omgezet zijn in declaraties. Met de taal DSL2 is het mogelijk om zowel synchrone als asynchrone systemen te beschrijven en te ontwerpen. De taal beschrijft het systeem door aan te geven welke signalen hoog en laag worden, maar ze bevat ook elementen van andere methoden om digitale systemen te beschrijven. Soms wil men liever een logische functie door zijn functietabel beschrijven. Op deze manier worden er functietabellen in deze taal toegestaan. Voor sommige systemen is het gemakkelijker om de gates en flip-flops te specificeren dan de procedures van het systeem (b.v. voor geheugens). Om deze redenen zijn er reële circuits met hun ingangen toegestaan. De taal DSL2 werd ontwikkeld door digitale systemen in verkorte vorm te schrijven. Bij deze beschrijving werden zoveel mogelijk de betekenis voor de acties gebruikt in de taal. Dit leverde een assembler georiënteerde taal op (DSL) welke werd uitgebouwd tot DSL2.
De gekozen taal heeft de volgende hoofdkenmerken:

1- Het systeem wordt beschreven in procedures en declaraties. Een systeembeschrijving mag verschillende procedures en declaraties bevatten.
2- Alle procedures worden parallel uitgevoerd.
3- Een statement in een procedure mag meer dan een instructie bevatten. Dit betekent dat deze instructies worden uitgevoerd onder dezelfde condities en op hetzelfde moment.
4- De instructies zijn gemakkelijk te veranderen en het is gemakkelijk om er nieuwe instructies aan toe te voegen.
5- Het is mogelijk om reële circuits te beschrijven met hun

onderlinge verbindingen zodat het gemakkelijk is om ieder digitaal systeem te beschrijven.

6- In principe is het mogelijk om macro-instructies te gebruiken welke worden vervangen door een set van normale instructies en die reeds beschreven systemen insluiten.

Om de uitvoering van de beschrijving eenvoudig te houden, worden de verschillende instructies en declaraties geconstrueerd vanuit prototypes. Een prototype bevat vaste en variabele velden. Bij het ontwerpen van een instructie of declaratie wordt het gewilde prototype gezocht. Daarna worden de vaste velden gecopiëerd en de variabele velden gevuld met de gewenste variabelen. Met deze methode is een lijst van prototypes nodig. Als een bepaalde instructie veranderd moet worden, wordt alleen het prototype veranderd. Als een nieuwe instructie aan de taal moet worden toegevoegd wordt er een nieuw prototype toegevoegd aan de lijst van prototypes.

De beschrijving wordt zo kort mogelijk gehouden door de invoering van indexes, macro instructies en een source statement library. Deze bibliotheek bevat de beschrijving van subsystemen. Bij het uitvoeren van het systeem kunnen subsystemen worden gecopiëerd vanuit de bibliotheek.

Als een systeem wordt beschreven door zijn componenten en zijn verbindingen, is het niet meer nodig om het systeem te ontwerpen. Een computer kan het systeem nog simuleren, de bedradingslijst genereren en de benodigde informatie en documentatie leveren. Deze mogelijkheid is noodzakelijk omdat een ontwerper altijd een systeem kan ontwerpen welke niet of nauwelijks in een taal beschreven kan worden, onafhankelijk van de taal welke werd gekozem. De stappen om het systeem in zo'n geval te realiseren kunnen als volgt worden samengevat:

1- globale beschrijving van het systeem; het te ontwerpen systeem is bekend als een blackbox waarvan de specificaties naar de buitenwereld duidelijk vastliggen.

2- detaillering van het systeem; de blackbox wordt verdeeld in een aantal bouwstenen zoals tellers, schuifregisters, combinatorische netwerken en zenders en ontvangers voor de verbindingen met de buitenwereld. Hierbij worden zoveel

mogelijk standaardschakelingen gebruikt

3- controle van het ontwerp; dit geschiedt door simulatie van de systeembeschrijving met een computer. Om de simulatietijd klein te houden moeten kleine subsystemen worden gesimuleerd
4- detaillering van de bouw; positionering van de bouwstenen
5- het bouwen; genereren van de bedradingslijst
6- testen
7- documentatie

Voor het automatisch ontwerpen van digitale systemen moet de gedragsbeschrijving in DSL2, worden omgezet in een beschrijving die geschikt is voor hardware uitvoering. De ontwerpprocedure bevat de volgende stappen:

1. Het gedrag van het systeem wordt beschreven in DSL2 d.w.z. via de geformaliseerde systeembeschrijving moet worden vastgelegd wat het systeem moet doen.
2. Uit de systeembeschrijving wordt het systeem ontworpen. Ontwerpen betekent in dit geval de omzetting van de procedures in declaraties. Hiervoor worden de executievoorwaarden van de instructies van de procedures vastgesteld en de logische vergelijkingen, welke aan deze eisen voldoen, gegenereerd. Als de procedures niet voldoende elementen beschrijven om aan alle eisen te kunnen voldoen, worden er extra elementen geïntroduceerd. De gehele ontwerpprocedure geschiedt op een conversationele manier, d.w.z. de ontwerper moet de resultaten van de computer beoordelen en eventueel de systeembeschrijving veranderen en extra elementen toevoegen.
   Daarna kan door de computer het systeem opnieuw worden gegenereerd.
3. Het nu verkregen ontwerp in Boolse functies moet worden omgezet in reële circuits met hun inputs.
   De ontwerper moet hierbij specificeren welke logische familie hij wil gebruiken. Als de realisatie wordt uitgevoerd met een computer, dan moet de computer een lijst produceren van alle bouwstenen met hun verbindingen.
4. Tot nu toe zijn vertragingen en andere niet ideale mogelijkheden niet beschouwd. Deze mogelijkheden kunnen worden

nagegaan door simulatie van het ontworpen systeem. De ontwerper voert inputpatronen toe en het simulatie-programma simuleert de toestanden van gewenste punten in het ontwerp. De toestanden van de punten worden als functie van de inputs geprint.

5. Na de simulatie moet het systeem worden gedetailleerd voor de bouw. Circuits moeten worden geplaatst op de circuit-boards, de pennummers moeten worden toegekend aan de input/ outputsignalen. Dit kan zelf worden gedaan of geheel of gedeeltelijk met de computer.
6. De computer levert nu de gedetailleerde systeembeschrijving. De beschrijving bevat alle bouwstenen, hun positie op de circuitboards en hun inputsignalen. Nu kan de bedradingslijst worden gegenereerd. De bedradingslijst kan eventueel in tape geponst worden om een bedradingsmachine te sturen.
7. Tenslotte kunnen programma's een geponste tape leveren voor het automatisch testen van het gebouwde systeem. De tape bevat de inputpatronen en de gewenste outputpatronen. De inputpatronen worden toegevoerd aan het systeem en de outputpatronen worden vergeleken met de outputpatronen op de tape. Verschillen worden gesignaleerd.

De taal DSL2 voldoet als ontwerptaal aan alle eisen welke in het begin van dit hoofdstuk werden genoemd. In vergelijking met andere talen bevat DSL2 enkele kenmerken die niet eerder in de literatuur zijn vermeld:

1. DSL2 is de enige taal die onafhankelijke procedures kent. De koppeling tussen de procedures is op dezelfde wijze opgelost als het in hardware gebeurt.
2. DSL2 is ontwikkeld vanuit de definitie van een digitaal systeem. Er bestaat dan ook een eenduidig verband tussen een DSL2- beschrijving en een digitaal systeem. Dit verband is bij geen enkele taal zo duidelijk aanwezig.
3. Een systeembeschrijving in DSL2 leent zich goed voor omzetting in een toestands- en outputtabel.

Enkele kritische opmerkingen:

1. In de practijk blijkt dat DSL2 het meest geschikt is voor

de beschrijving en het ontwerp van kleine systemen. Dit zijn systemen welke niet kunnen worden verdeeld in subsystemen, maar direct in gates, flipflops en andere circuits. In principe kan DSL2 ook gebruikt worden voor het beschrijven en ontwerpen van grote systemen.

2. Er zijn voor het beschrijven van digitale systemen talen die qua taal beter zijn dan DSL2.
3. De gebruiker moet voldoende kennis van de ontwerpmethode hebben bij de eerste implementaties om het DSL2-systeem te kunnen gebruiken.

3. BESCHRIJVING VAN DIGITALE SYSTEMEN MET DSL2.

3.1 Definiëring van een digitaal systeem.

Voordat over de beschrijving van digitale systemen gesproken kan worden moet eerst worden gedefinieerd wat een digitaal systeem is. Uit de eigenschappen van een digitaal systeem kan de volgende definitie worden afgeleid:

Definitie 3.1.1:

Een digitaal systeem is het 6-tuple M:

M = (S,T,I,O,d,l)

met:

- S: een eindige verzameling toestanden
- T: een verzameling initiele toestanden. T is een deelverzameling van S
- I: een eindige verzameling inputs
- O: een eindige verzameling outputs
- d: een volgende toestand functie. Dit is een partiele afbeelding van S en I op S
- l: een output functie. Dit is een afbeelding van S op O of een partiele afbeelding van S en I op O.

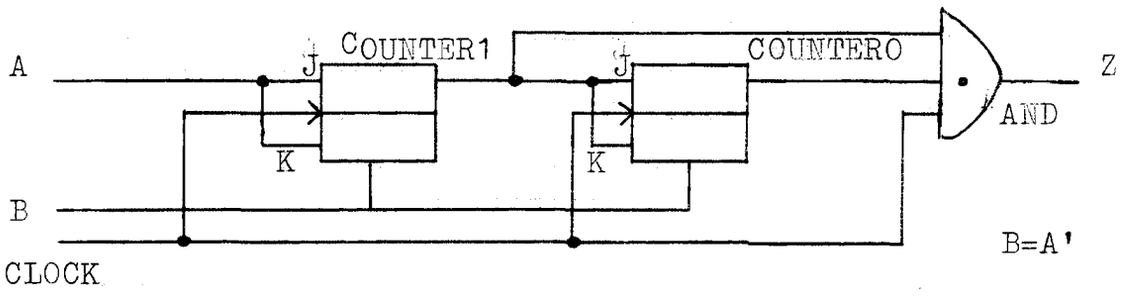
De functies d en l zijn partiele functies als er bij een combinatie van toestand en input geen volgende toestand, respectievelijk output is gedefinieerd.

Bij het beschrijven van digitale systemen dienen deze zes eigenschappen dus eenduidig beschreven te worden.
Een mogelijke vorm van deze beschrijving is de sequentietabel. Figuur 3.1.1 geeft hiervan een voorbeeld.

	a	b	a	b
1	2	1	0	0
2	3	1	0	0
3	4	1	0	0
4	1	1	1	0

Figuur 3.1.1: sequentietabel van een vierteller.

De eerste kolom van figuur 3.1.1 beschrijft de mogelijke toestanden (S). De eerste rij beschrijft de input (I). De tweede en derde kolom beschrijven de volgende toestand functie d, terwijl de laatste 2 kolommen de outputfunctie l beschrijven. De outputelementen worden gegeven door de gebruikte symbolen in deze kolommen. In dit geval is de output een functie van de toestand en de input. Als de output alleen een functie van de toestand is kan deze in een kolom worden geschreven. De verzameling initiele toestanden (T) is een deelverzameling van (S). De schakeling beschrijft een vierteller. De teller telt tot 4 als input a aanwezig is. Als de input b is wordt de teller gereset in toestand 1. Als de teller stand 4 bereikt, en de input a is nog steeds aanwezig, dan wordt de output 1 gegenereerd. Als deze tabel een reëel systeem moet beschrijven, moet worden aangenomen dat er nog een klok is die de toestandsveranderingen bestuurt. Als een dergelijke klok niet verondersteld wordt en hetzelfde systeem moet in een tabel beschreven worden dan wordt de tabel aanmerkelijk groter.
Figuur 3.1.2 geeft de schakeling van de vierteller beschreven in figuur 3.1.1



Figuur 3.1.2: Gesynchroniseerde binaire vierteller.

Het systeem bestaat uit 2 JK-flipflopppen, COUNTER1 en COUNTERO een AND poort en de benodigde input/output verbindingen. De flipfloppen veranderen van stand als de J en K ingangen hoog zijn en de klok-ingang hoog wordt. Als de R-ingang hoog wordt, wordt de flipflop gereset in toestand 0. Het systeem is een voorbeeld van een level-input, level-output, clocked sequential circuit (LLC circuit(26)).

Ook door het geven van het type systeem kunnen een aantal eigenschappen impliciet beschreven worden. Zo heeft de AND poort uit figuur 3.1.2 als de vertraging verwaarloosd wordt, slechts 1 toestand. Dit is tevens de initiele toestand. De verzameling outputs bevat per uitgang twee elementen, nl. output hoog (1) en output laag (0). De volgende toestand functie is triviaal (s=d(s)). De outputfunctie wordt gegeven door de configuratie van de poort. De output van de AND poort is alleen dan hoog als alle inputs hoog zijn. De eindige verzameling inputs (I) wordt bepaald door het aantal ingangen van de poort. Een poort met n ingangen heeft 2^n inputs. Het werkelijk aantal inputs wordt bepaald door het aantal gebruikte ingangen en de relatie tussen de ingangssignalen. Er blijkt dus dat de poort als bouwsteen ook een digitaal systeem beschrijft want alle grootheden van definitie 2.1.1 zijn aanwezig. Voor de JK flipflop geldt eenzelfde beschrijving. De flipflop beschrijft ook een aantal eigenschappen impliciet. De flipflop heeft twee toestanden (S) met de initiele toestand als deelverzameling van S. De verzameling outputs bevat per uitgang twee elementen, nl. output hoog (1) en output laag (0). De twee uitgangen van de flipflop zijn elkaars inverse. De J en K ingang zorgen voor een eindig aantal inputs. De volgende toestand functie en de output functie kunnen afgeleid worden uit figuur 3.1.3. Volgens definitie 2.1.1 is de JK-flipflop dus een digitaal systeem.

	a	b	c	d	a	b	c	d
1	1	1	2	2	0	0	1	1
2	2	1	2	1	1	0	1	0

Figuur 3.1.3: Sequentietabel van een JK flipflop.

Uit het voorbeeld van figuur 3.1.2 blijkt dat een digitaal systeem kan bestaan uit een aantal subsystemen waarbij ieder subsysteem weer een digitaal systeem beschrijft. Ook blijkt dat een aantal digitale systemen die via een verbindingsnetwerk onderling en met de inputs en outputs gekoppeld worden, weer een digitaal systeem beschrijven.

Hoewel definitie 3.1.1 ieder digitaal systeem definieert, kunnen

sommige digitale bouwstenen beter worden gespecificeerd door de toestanden, inputs en outputs te representeren door binaire variabelen. Dit levert de volgende definitie:

Definitie 3.1.2:

De binaire input variabelen: $x(i)$ met $1 \leq i \leq m$

de binaire toestand variabelen: $y(i)$ met $1 \leq i \leq n$

de overgangsfuncties: $Y(i); \{y(1),...y(n),x(1),...x(m)\} \rightarrow \{0,1\}$ met $1 \leq i \leq n$

de output functies: $z(k): \{y(1),...y(n),x(10,...x(m)\} \rightarrow \{0,1\}$ met $1 \leq k \leq r$

definieren het systeem

$$M = (S,T,I,O,d,l)$$

met $S = \{(y(1),....y(n))\}$ de verzameling n-tuples op $\{0,1\}$

T = een deelverzameling van S

$I = \{x(1),.....x(m)\}$

$O = \{(z(1),....z(r))\}$

$d = Y$

$l = z$

Een voorbeeld van deze definitie is gegeven in figuur 3.1.4.

$x_1,x_2 \rightarrow$ $y_1 \downarrow$	00	01	10	11	00	01	10	11
0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0

Figuur 3.1.4: Voorbeeld van een systeem waarbij de toestanden, inputs en outputs worden gerepresenteerd door binaire variabelen, respectievelijk y(1), x(1), x(2), z(1).

Figuur 3.1.4 beschrijft hetzelfde systeem als figuur 3.1.3. De tabel kan ook als volgt worden genoteerd (zie figuur 3.1.5)

toestand	vorige toestand	input x(1)	x(2)	output z
0	0	0	0	0
1	1	0	0	1
0	0	0	1	0
0	1	0	1	0
1	0	1	0	1
1	1	1	0	1
1	0	1	1	1
0	1	1	1	0

Figuur 3.1.5: Andere notatie van de tabel uit figuur 3.1.4.

Uit figuur 3.1.5 blijkt dat zowel met input 00 als met input 01 toestand 0 overgaat in toestand 0. Bij beide overgangen is de output 0. Dit kan worden samengevat met te zeggen dat bij input 0X toestand 0 overgaat in toestand 0, waarbij de toestand van de variabele als x genoteerd wordt als deze 0 of 1 kan zijn. Verder gaande met deze notatie kan figuur 3.1.5 omgewerkt worden tot figuur 3.1.6.

toestand	vorige toestand	input		output
0	0	0	X	0
0	1	X	1	0
1	0	1	X	1
1	1	X	0	1

Figuur 3.1.6: Andere notatie van de tabel uit figuur 3.1.5.

Wordt nu de outputvariabele zodanig gekozen dat deze overeenkomt met de toestandsvariabele, dan komen de eisen voor de toestandsvariabele overeen met de eisen van de outputvariabelen. Figuur 3.1.7 geeft de tabel waaruit de logische vergelijkingen van de JK flipflop gemakkelijk kunnen worden afgeleid.

x(1)=J	x(2)=K	z_k	z_{k+1}
0	X	0	0
X	1	1	0
1	X	0	1
X	0	1	1

Figuur 3.1.7: JK-flipflop met waarheidstabel.

Uit figuur 3.1.7 blijkt voor de onderhavige flipflop te gelden: $z_{k+1} = z_k \cdot K' + J \cdot z_k'$

In het voorgaande is gebleken dat een digitaal systeem in de praktijk wordt opgebouwd uit een aantal subsystemen die via een verbindingdnetwerk onderling en met de inputs en outputs gekoppeld worden. Door het beschrijven van de subsystemen, inputs, outputs en verbindingen wordt dus ook een digitaal systeem beschreven. Deze constructie wordt een netwerk genoemd en wordt als volgt gedefinieerd:

Definitie 3.1.3.

Een netwerk N van subsystemen bestaat uit:

1. $\{M(i) = (S(i),T(i),I(i),O(i),d(i),l(i))\}$ met $1 \leq i \leq n$ een aantal subsystemen
2. Een eindig aantal inputs I
3. Een eindig aantal outputs O
4. Verbindingsfuncties f(i): $(*O(j))*I \rightarrow I(i)$ met $1 \leq j \leq n$ [1)]
5. Een output functie l: $(*O(j))*I \rightarrow O$ met $1 \leq j \leq n$

Volgens deze definitie dient de beschrijving van een netwerk dus te bestaan uit de beschrijving van:

1. De gebruikt subsystemen
2. De netwerk-inputs
3. De netwerk-outputs
4. De inputs van de subsystemen als functies van de netwerk-inputs en outputs van de subsystemen.
5. De netwerk-outputs als functie van netwerk-inputs en outputs van subsystemen.

Uiteraard kan dit netwerk weer als systeem worden opgevat.

Definitie 3.1.4. Een netwerk N definieert het systeem M:

$$M = (S,T,I,O,d,l)$$

met:

$$S: * S(i) \quad \text{met} \quad 1 \leq i \leq n$$

$$T: * T(i) \quad \text{met} \quad 1 \leq i \leq n$$

1) Stel S en T zijn niet lege verzamelingen. De functie f:
f: S -> T
definieert bij iedere s in S een t in T. Notatie: f(s)=t.
Het cartesisch product van een aantal verzamelingen S1,...Sn is de verzameling van alle n-tuples (s(1),...s(n)) met s(i) in Si, geschreven als:

$$S1 * S2 * \ldots * Sn = * Si = \{(s(1),\ldots s(n)) \mid s(i) \text{ in } S(i)\}$$

I: inputs I van N
O: outputs O van N
d: *d(i) met 1≤i≤n
l: output functie l van N

Volgens deze definities kan een systeembeschrijving gestructureerd zijn. Een aantal elementaire systemen wordt als bouwsteen in een netwerk gebruikt. Dit netwerk wordt als subsysteem van een omhullend netwerk gezien. Een eventuele structuur in de beschrijving heeft echter geen enkele invloed op de koppeling tussen de verschillende subsystemen. De koppeling bestaat alleen uit de gespecificeerde verbindingsfuncties. Verder zijn de subsystemen onafhankelijke onderdelen, zodat de volgorde van de beschrijvingen van de subsystemen geen enkele invloed heeft op de werking van het geheel.

3.2 Structuur van een DSL2 beschrijving.

In hoofdstuk 2 zijn reeds een aantal hoofdkenmerken van DSL2 genoemd. Voor een volledige beschrijving van DSL2 wordt verwezen naar de appendix van literatuuropgave 1.
De structuur van een DSL2 beschrijving kan het beste worden geïllustreerd aan de hand van enkele voorbeelden. Hierbij zal blijken dat er een eenduidig verband bestaat tussen een DSL2 beschrijving en een digitaal systeem.

Met DSL2 is het mogelijk om bestaande systemen te beschrijven. Als voorbeeld zal de gesynchroniseerde binaire vierteller uit figuur 3.1.2 worden beschreven.

```
1  SYSTEM;
2      COUNTER1:        JKFF          J = A
                                      K = A
                                      C = CLOCK
                                      R = B;
3      COUNTER0:        JKFF          J = COUNTER1
                                      K = COUNTER1
                                      C = CLOCK
                                      R = B;
4      C:               AND           I = COUNTER0,COUNTER1,A;
5      C:               OUTPUT;
6      A:               INPUT;
7      B:               INPUT = A';
8      CLOCK:           INPUT;
9  END;
```

Programma 1.

Deze beschrijving beschrijft een netwerk. Alle elementen die volgens definitie 3.1.3 aanwezig moeten zijn zijn aanwezig. De subsystemen zijn COUNTER0, COUNTER1 en C. De inputs zijn A en B. De output is C. De verbindingsfuncties zijn verwerkt in de beschrijving van de subsystemen. Bij ieder subsysteem wordt een verbindingsfunctie beschreven. De output-functie wordt beschreven door de dubbele functie van C.
De beschrijving van de subsystemen voldoet aan de definities 3.1.1 en 3.1.2. De elementen S,T,0,d, en 1 zijn impliciet beschreven, I expliciet.
Het beschrijven van een bestaand digitaal systeem kan in DSL2 geschieden door het systeem op te splitsen in een aantal subsystemen. Deze subsystemen worden weer beschreven in een aantal echte bouwstenen, die parallel kunnen werken. De input/output beschrijving en eventueel commentaar complementeren de DSL2 beschrijving.

Voor het beschrijven van te ontwerpen systemen kan de zojuist aangegeven methode niet worden gebruikt. Bij het begin van het ontwerpen van systemen zijn in het algemeen slechts enkele bouwstenen en verbindingen bekend. Daarnaast is echter ook bekend wat het systeem moet doen. Om het ontwerpproces zo eenvoudig en efficient mogelijk te houden zal alle bekende informatie in de beschrijving opgenomen moeten worden. Het beschrijven van de bouwstenen kan hierbij op de hiervoor geschetste manier geschieden. Als de te gebruiken bouwstenen nog niet exact bekend zijn moeten symbolische bouwstenen worden gebruikt. Om op eenvoudige wijze onderscheid te kunnen maken tussen 'wat het systeem moet doen' m.a.w. het gedrag van het systeem en de beschrijving van de bouwstenen, kan de gedragsspecificatie het beste als speciaal subsysteem worden beschreven. Dit subsysteem heeft dan een aantal verbindingen met de beschreven bouwstenen. De gedragsspecificatie kan worden beschreven in een algorithme. Het algorithme is in DSL2 geformaliseerd in een procedure. Omdat de procedure dus een subsysteem beschrijft zal de beschrijving moeten voldoen aan een van de definities 3.1.1/3.1.3. Dit zal worden nagegaan aan de hand van een voorbeeld.

```
1     SYSTEM;
2        PROCEDURE;
3             CLOCK IS 0;
4             WAIT FOR CLOCK = 1;
5             IF A = 1 THEN COUNT COUNTER;
6                      ELSE RESET COUNTER;
7             WAIT FOR CLOCK = 0;
8        END;
9        COUNTER:     COUNTER LENGTH = 2;
10       C:           =COUNTER(0).COUNTER(1).A;
11       C:           OUTPUT;
12       A,CLOCK:     INPUT;
13    END;
```

Programma 2

In het voorbeeld worden de te verrichten acties beschreven in de procedures. De voornaamste bouwstenen en de inputs en outputs vormen de rest van de beschrijving.

De procedure is opgebouwd uit een aantal statements. De statements worden achtereenvolgens verwerkt. De statements worden verdeeld in acties, besturingsstatements en beschrijvende statements. De acties beschrijven de uit te voeren instructies. Besturingsinstructies beschrijven, al dan niet conditioneel, het volgende te verwerken statement. Als het volgende te verwerken statement niet het volgende statement in de procedure is, dient dit gespecificeerd te zijn. Het eerste statement wordt beschouwd als het volgende statement van het laatste statement van een procedure. Deze constructie heeft tot gevolg dat als de procedure verwerkt is, automatisch opnieuw met de verwerking wordt begonnen; dit in analogie met een digitaal systeem. Beschrijvende statements beschrijven de toestand van een aantal inputs van de procedure. De beschreven toestanden dienen correct te zijn als het beschrijvende statement verwerkt wordt.
In het voorbeeld wordt een asynchroon systeem beschreven want er wordt niet aangegeven dat de toestand veranderingen op commando van een klok-signaal plaats vinden. Het programma geeft een asynchrone beschrijving van de binaire vierteller. De statements uit de procedure spreken voor zich. In de procedure zijn de bouwstenen COUNTER en C reeds bekend. Beide zijn op-

genomen in de beschrijving als symbolische bouwstenen. COUNTER moet een twee bit teller zijn en C een combinatorisch netwerk. Bij de beschrijving van C is gebruik gemaakt van het feit dat door het specificeren van COUNTER als 2 bit teller de naam COUNTER geïndexeerd kan worden om de variabelen van de teller aan te duiden. Hierbij is de laagste index 0. Deze beschrijft de meest significante variabele. Op de realisatie van het combinatorisch netwerk wordt in de beschrijving niet ingegaan. Tenslotte zijn de systeem-inputs en systeem-outputs bekend. Voor de teller is aangenomen dat de reset stand voor de teller 0 is.

Vergelijken we de procedure met de definitie van een digitaal systeem (definitie 3.1.1) dan blijken alle grootheden uit de definitie aanwezig te zijn. De verzameling S kan gerepresenteerd worden door de verzameling statements. T wordt gerepresenteerd door het eerste statement van de procedure. I kan alle elementen die in de procedure genoemd worden bevatten. De instructies beschrijven acties die door andere subsystemen uitgevoerd worden. O kan dan de verzameling signalen die deze instructies uitvoeren representeren. De functies d en l zijn in de procedure verweven.

Uit het voorbeeld blijkt dat een procedure beschouwd kan worden als een subsysteem dat via een verbindingsnetwerk met andere subsystemen, bouwstenen, inputs en outputs verbonden is.

De asynchrone binaire vierteller kan ook als synchroon netwerk worden beschreven. Hiertoe wordt in het procedure statement een klok-signaal gespecificeerd omdat dit klok-signaal voor het gehele (sub)systeem geldt. Alle toestand veranderingen vinden nu plaats op commando van dit klok-signaal. De statements 2/8 van programma 2 moeten worden vervangen door de volgende statements:

```
2    PROCEDURE (CLOCK = CLOCK);
3          IF A = 1 THEN COUNT COUNTER;
4                   ELSE RESET COUNTER;
5    END;
```

PROGRAMMA 3

Ook voor programma 3 blijken de eigenschappen van de procedure overeen te komen met de eigenschappen van een digitaal systeem.

Tussen programma 1 en 2,3 zijn enkele essentiele verschillen. Een subsysteem beschreven in bouwstenen bevat een aantal bouwstenen die allen tegelijk actief zijn. De volgorde van beschrijving heeft geen enkele invloed op de werking van het subsysteem. Dit is niet het geval bij de procedure. De beschrijving van de procedure bestaat uit statements. Verandering van de volgorde van deze statements geeft een andere procedure. Ook kan, door het verschil in betekenis tussen procedures en subsystemen opgebouwd uit bouwstenen, de procedure geen beschrijving van bouwstenen bevatten.

Definitie 3.2.1
Een procedure P is het 6-tuple:

$$P=(S,T,I,O,d,l)$$

met S: een eindige niet lege verzameling statements
T: het eerste statement van de procedure
I: een eindige niet lege verzameling inputs
O: een eindige niet lege verzameling outputs
d: een volgende statement functie. d is een partiele afbeelding van S en I op S
l: een output functie. l is een afbeelding van S op O

De functie d is een partiele functie omdat het volgende statement niet voor alle combinaties van statement en output gedefinieerd is.
Er is geen wezenlijk verschil tussen definitie 3.1.1 en 3.2.1 zodat er geen bezwaar tegen is om de procedure als subsysteem te gebruiken.

Het feit, dat een procedure een subsysteem specificeert, heeft de volgende consequenties:

1. een procedure kan niet gestopt worden, omdat de procedure een digitaal systeem beschrijft. Daarom wordt het eerste statement van de procedure gedefinieerd als het volgende

statement van de laatste statement

2. Een procedure mag geen andere procedure bevatten; een procedure vertegenwoordigt een onafhankelijk subsysteem
3. Om dezelfde reden kan niet van de ene procedure naar de andere procedure worden gesprongen.
4. Procedures kunnen parallel worden verwerkt d.w.z. op hetzelfde moment worden uitgevoerd.
5. Omdat onafhankelijke subsystemen kunnen worden gekoppeld moeten procedures ook kunnen worden gekoppeld. Dit kan gerealiseerd worden door wachtinstructies.

Voorbeeld:

```
1     SYSTEM;
2     X: PROCEDURE;
3             STATE OF A IS 0;
4             ....;
5             ....;
6             SET A;
7             WAIT FOR A IS 0;
8             ....;
9             ....;
10            ....;
11    END;
12    Y: PROCEDURE;
13            STATE OF A IS 0;
14            WAIT FOR A IS 1;
15            ....;
16            ....;
17            ....;
18            RESET A;
19    END;
20    A: FLIPFLOP;
21    END;
```

Programma 4

Dit systeem bestaat o.a. uit de procedures X en Y en de flip-flop A. Beide procedures beginnen met te stellen dat de toestand van A 0 is. A kan bv. bij het inschakelen gereset worden. Procedure Y begint dan te wachten tot de toestand van A 1 is. Procedure Y wordt normaal verwerkt. In statement 6 wordt de toestand van A op 1 gezet. Daarna wacht procedure X tot de toestand van A weer 0 is. Zodra de toestand van A 1 is geworden kan procedure Y verder uitgevoerd worden. In statement 18 wordt de toestand van A weer 0. Daarna wordt het

einde van de procedure bereikt. Als volgende statement wordt nu statement 13 verwerkt. Daarna wacht procedure Y weer tot de toestand van A 1 is geworden. Zodra de toestand van A 0 wordt, wordt ook de uitvoering van procedure X voortgezet.

Door deze constructie start procedure X de procedure Y en wacht tot procedure Y verwerkt is. Daarna wordt de uitvoering van procedure X voortgezet. Uit de beschrijving blijkt echter ook dat beide procedures in wezen onafhankelijk van elkaar zijn. Zij worden gekoppeld via het element A.

Een systeembeschrijving van meerdere subsystemen kan in DSL2 een structuur bezitten. Deze structuur doet echter geen afbreuk aan de parallelle werking van de onafhankelijke subsystemen. De structuur wordt verduidelijkt door het gebruik van labels. Het is mogelijk om de elementen van een systeem, elementaire bouwstenen, subsystemen, inputs en outputs, aan te duiden. Dit kan geschieden door deze elementen een naam te geven. Een element wordt dan aangeduid door diens naam op te geven. Hierbij moet wel worden vastgelegd binnen welk gebied een bepaalde naam geldig is. Het gebied waarin een naam geldig is kan uit definitie 3.1.3 worden afgeleid. Volgens definitie 3.1.3 wordt een systeem opgebouwd uit een aantal subsystemen. De verbindingen tussen de subsystemen worden beschreven in het omhullende (sub)systeem. Deze verbindingen zijn verbindingen tussen de inputs en outputs van de subsystemen. Uit deze beschouwing volgt dat een omhullend systeem niet weet wat een subsysteem bevat maar alleen diens inputs en outputs kent. Dit betekent dat een naam alleen bekend is binnen het (sub)systeem waarin de naam gedefinieerd is. Namen die als input of output gedefinieerd worden zijn bovendien bekend in het omhullende systeem.

In het volgende voorbeeld bestaat het systeem A uit de subsystemen B en C. De input van A is P en de output is Q. Het subsysteem B bevat de procedure X, input R en output S. Het subsysteem C bevat de procedure Y, het subsysteem D, de input T en de output U. Tenslotte bevat het subsysteem D de procedure Z, de input V en de output W.

Voorbeeld:

```
A:  SYSTEM;
      B: SYSTEM;
           X: PROCEDURE;
                .
                .
              END;
           R: INPUT;
           S: OUTPUT;
         END;
      C: SYSTEM;
           Y: PROCEDURE;
                .
                .
              END;
           D: SYSTEM;
                Z: PROCEDURE;
                     .
                     .
                   END;
                V: INPUT;
                W: OUTPUT;
              END;
           T: INPUT;
           U: OUTPUT;
         END;
      P: INPUT;
      Q: OUTPUT;
    END;
```

Welke namen in welk (sub)systeem bekend zijn is beschreven in onderstaande tabel:

systeem	bekende namen
A	B,C,P,Q,R,S,T,U
B	R,S,X
C	D,T,U,V,W,Y
D	V,W,Z

Uit deze tabel en de voorgaande beschouwing volgt dat de naam van een (sub)systeem niet bekend is binnen dit systeem. Deze is bekend in het omhullende systeem.

Samenvatting

Moet een systeem worden ontworpen d.w.z. kennen we niet alle bouwstenen en verbindingen maar weten we wel wat het systeem moet doen, dan zijn voor het beschrijven van het digitale systeem de volgende elementen noodzakelijk:

a. Procedures
b. Symbolische bouwstenen

c. Echte bouwstenen
d. Input/output beschrijvingen
e. Verbindingen tussen de elementen

Met deze elementen worden subsystemen beschreven. De subsystemen worden verenigd in de beschrijving van het omhullend systeem. De subsystemen kunnen allen parallel werken.

Een systeembeschrijving van een bestaand digitaal systeem bevat geen procedures en symbolische bouwstenen. Het systeem wordt opgesplitst in een aantal subsystemen. Deze subsystemen worden weer beschreven in een aantal echte bouwstenen, die parallel kunnen werken. Verder bevat de beschrijving ook input/output beschrijvingen en verbindingen tussen de elementen.

De beschrijving van een willekeurig systeem in DSL2 is een verzameling beschrijvingen van subsystemen, bouwstenen en procedures. Ieder subsysteem, bouwsteen en procedure beschrijft weer een digitaal systeem. De verbindingen tussen de verschillende onderdelen worden beschreven als aparte verbindingen of worden opgenomen in de beschrijving van de onderdelen. Daarnaast bevat de systeembeschrijving nog een beschrijving van de systeem-inputs en de systeem-outputs. Door de aanwezigheid van commentaar in de systeembeschrijving kan de werking van het systeem duidelijk worden gemaakt.

3.3 Beschrijving van de algorithmes.

Zoals reeds is vermeld in 3.2 zal het gedrag van het systeem beschreven worden in een algorithme, geformaliseerd in een procedure. Ook is aangegeven dat een procedure een subsysteem beschrijft. Dit subsysteem wordt, evenals andere subsystemen, via een verbindingsnetwerk met de andere subsystemen, bouwstenen, inputs en outputs verbonden. Door deze constructie worden de mogelijkheden om te beschrijven 'wat het systeem moet doen' beperkt. Uit de constructie volgt dat het volgende beschreven kan worden:

a. Toestanden van input-variabelen.
b. Overgangen naar andere toestanden van het subsysteem.

c. Toestanden van de output-variabelen.

De mogelijkheden b en c kunnen afhankelijk van de toestand van de input zijn.

Een procedure wordt opgebouwd uit een aantal statements welke sequentieel worden verwerkt. Een statement wordt opgebouwd uit een of meerdere instructies, die parallel worden uitgevoerd. De mogelijkheden a/c worden in DSL2 beschreven met 3 soorten instructies:

1. Beschrijvende instructies welke de toestanden ván de input-variabelen beschrijven.
   De algemene vorm: STATE OF (verzameling) IS (toestand).
   Hierbij beschrijft (verzameling) de input-variabelen, terwijl (toestand) 0,1 of X (don't care) kan zijn.
   Om verschillende toestanden te beschrijven kunnen meerdere instructies in een statement worden opgenomen.
2. Besturingsinstructies. Deze beschrijven op een of andere wijze het volgende te verwerken statement, d.w.z. de volgende toestand. Hierdoor wordt een zekere structuur in de procedure gebracht.
   Dit kan gebeuren door het gebruik van DO- en END-statements. De statements tussen een DO-statement en een END-statement vormen een routine.
   DSL2 kent de volgende routines:

```
routine zonder conditie:          DO;
                                  ....
                                  ....
                                  END;
routine met start conditie:       IF (bewering) THEN DO;
                                  ....
                                  ....
                                  END;
                                  ELSE DO;
                                  ....
                                  ....
                                  END;
routine met uitvoeringsconditie:  DO WHILE (bewering);
                                  ....
                                  ....
                                  END;
routine met stop conditie:        DO UNTIL (bewering);
                                  ....
                                  ....
                                  END;
```

Verder bestaan nog de besturingsinstructies voor de niet-conditionele en conditionele sprongen:

niet-conditionele sprong	GO TO (label)
conditionele sprong	IF (bewering) THEN (instr 1) ELSE (instr 2)

Voor de sprongopdrachten moeten de statements gelabeld zijn.

3. Acties om toestanden van output-variabelen te beschrijven.
Via het verbindingsnetwerk (definitie 3.1.3) besturen de outputs van de procedure de inputs van de andere subsystemen en bouwstenen. Door nu in de procedures de inputs van de te besturen bouwstenen te beschrijven, wordt naast de outputs van de procedure, tevens een deel van het verbindingsnetwerk beschreven.
Voorbeelden van acties zijn:

SET (set)
RESET (set)
LOAD (set)
COUNT (set)
SHIFT (set)
RAISE (set)
DROP (set)
WAIT FOR (symbol) IS (state)

In deze voorbeelden beschrijft (set) de te bewerken bouwstenen.
De eerste acties uit deze lijst beschrijven duidelijk outputs van de procedures en ingangen van andere bouwstenen en subsystemen. Dit geldt niet voor de laatste actie. Deze actie beschrijft dat de verwerking van de procedure moet wachten tot de toestand van het gespecificeerde element de gespecificeerde waarde heeft.
Bij de genoemde acties beschrijft (set) de te bewerken bouwstenen of subsystemen. De 'operatie codes' kunnen voor de verschillende bouwstenen verschillende betekenissen hebben. De acties worden door de bouwstenen en subsystemen uitgevoerd, de procedure activeert alleen de betreffende input.
Omdat de beschrijving van de acties sterk afhankelijk is van de te bewerken bouwstenen en omdat er altijd nieuwe bouwstenen ontwikkeld zullen worden, moet het mogelijk zijn om nieuwe 'operatie codes' in te voeren.

3.4 Beschrijving van de bouwstenen.

Het beschrijven van de bouwstenen bestaat uit drie onderdelen:

1. beschrijving van reële circuits
2. beschrijving van de symbolische bouwstenen
3. beschrijving van de inputs en outputs

Voor wat de beschrijving betreft, kunnen de inputs en outputs goed tot bouwstenen worden gerekend. De input kan als bouwsteen worden beschouwd van het type INPUT zonder inputs terwijl de output beschouwd kan worden als bouwsteen van het type OUTPUT waarbij de output functie als input kan worden opgevat.

Voor het beschrijven van reële circuits dient het type bouwsteen en de input van de bouwsteen, indien bekend, beschreven te worden. Hierbij moet worden gestreefd naar een zo kort mogelijke schrijfwijze. De uitgangen van een circuit worden als input van andere circuits gebruikt zodat een circuit ook een naam moet hebben.

Voorbeeld 1:

```
COUNTER1:                JKFF        J=A
                                     K=A
                                     C=CLOCK
                                     R=B;
COUNTER0:                JKFF        J=COUNTER1
                                     K=COUNTER1
                                     C=CLOCK    (zie programma 1)
                                     R=B;        hoofdstuk 3.2
```

Voorbeeld 2:

```
FF1:                     MC1013      J=CLOCK
                                     K=CLOCK;
FF0:                     MC1013      J=FF1/N,CLOCK
                                     K=FF1/N,CLOCK;
```

Voorbeeld 3:

```
1    SHIFTREG0:          DDF         D=DATA
                                     C=CLOCK;
2    SHIFTREG1:          DDF         D=SHIFTREG0
                                     C=CLOCK;
3    SHIFTREG2:          DDF         D=SHIFTREG1
                                     C=CLOCK;
4    SHIFTREG3:          DDF         D=SHIFTREG2
                                     C=CLOCK;
```

De voorbeelden beschrijven reële circuits. Voorbeeld 3 beschrijft een 4-bit schuifregister. Deze notatie is echter voor grote schuifregisters veel te omslachtig. Voor een 32-bit schuifregister zou de beschrijving uitgroeien tot 64 regels. Een veel kortere beschrijving is de volgende:

```
1  SHIFTREGO:              DDF          D=DATA
                                        C=CLOCK;
2  .GENERATE FOR I=1 TO 31;
3  SHIFTREG&I              DDF          D=SHIFTREG&(&I-1)
                                        C=CLOCK;
4  .                       GEND;
```

Het nadeel van deze notatie is de aanwezigheid van het GENERATE algorithme. De volgende beschrijving heeft dit nadeel niet doch is moeilijker te verwerken:

```
1  SHIFTREG:    LENGTH=32, CIRCUIT=DDF, D-INPUT=DATA,
                C-INPUT=CLOCK;
```

Hierbij wordt het schuifregister als 'macro' gespecificeerd zodat er een macro-definitie en macro-processor nodig is om de beschrijving te genereren. Bij de eerste verkorte schrijfwijze worden a.h.w. macro-definitie en macro-instructie tegelijk gespecificeerd.
De tweede methode heeft de voorkeur als veel beschrijvingen met dezelfde structuur verwerkt worden. Is dit niet het geval dan verdient de eerste methode de voorkeur.
Als een circuit meerdere uitgangen heeft, worden deze onderscheiden door nummering.
Voorbeeld 4:
Outputpen 3 van het circuit GATE kan worden aangeduid met

GATE/3 of GATE/01

waarbij dient te worden gespecificeerd dat 01, 3 betekent.

Het ligt voor de hand om de naam zonder toevoeging te gebruiken, voor de enige of belangrijke uitgang van een circuit. Zo is de input van COUNTER0 in voorbeeld 1 de uitgang van COUNTER1 terwijl de inputs van FF0 in voorbeeld 2 bestaan uit de non-uitgang van FF1 (FF1/N) en de clock. Op deze manier worden de verbindingsfuncties in de beschrijving van het circuit opgenomen.

Als een circuit meerdere ingangen bezit, moeten deze kunnen worden onderscheiden. Dit kan geschieden door het aangeven van inputpennen. Als alle ingangen gelijkwaardig zijn (zoals van een poort), is het niet noodzakelijk om de inputpennen te beschrijven.

Voorbeeld 5:

```
GATE0:          MC1010      FF1,FF0;
GATE1:          MC1010      FF1/N,FF0/N;
```

Als er verschillende soorten ingangen zijn (zoals bij flip-floppen), dient de inputpen wel beschreven te worden. Dit kan bv. door de beschrijving van de input-pen tussen haakjes achter de beschrijving van de output te plaatsen.

Voorbeeld 6:

De twee gates uit voorbeeld 5 kunnen worden gecombineerd tot 1 gate, met bv. 6 ingangen. Dit zou de volgende beschrijving kunnen opleveren:

```
GATE:           MC1010      FF1(1),FF0(2),
                            FF1/N(5),FF0/N(3);
```

Deze methode vereist teveel werk als er meerdere gelijkwaardige ingangen van hetzelfde type zijn en als alleen maar het type ingang aangegeven moet worden. In dit geval is het prettig om het type ingang voor de inputsignalen aan te geven.

Voorbeeld 7:

```
BUSY:           FF          S=SET1,SET2
                            R=RESET1,RESET2;
```

Zie ook voorbeelden 1/3.

Zoals reeds is opgemerkt, kunnen de inputs en outputs, voor wat betreft de beschrijving, goed tot bouwstenen worden gerekend. De bouwstenen zijn van het type INPUT en OUTPUT.
Het 'output-circuit' dient een verbinding te beschrijven. Dit kan op twee manieren gebeuren, nl. door een input aan te geven zoals bij poorten gebeurt of door de naam van het'outputcircuit' gelijk te kiezen aan de naam van het circuit waar de verbinding vandaan komt.

Voorbeeld 8:

Een poort GATE0 heeft als uitgang TO. De uitgang TO is tevens output van het subsysteem. Dit kan worden beschreven als:

```
TO:             OUTPUT      =GATE0;
```

Boolse functies worden opgenomen, bv:

```
X=B+C;
Y=A.B+C.D;
Z=(A'+B).((C+D)'.E+F);
```

Ook is het mogelijk om combinatorische netwerken met functietabellen te beschrijven, bv:

```
X=F(0,1,4,7) of A,B,C;
```

De waarde van de toestand van de functie is 1 als de variabelen A, B en C de toestanden 000, 001, 100 of 111 hebben. In plaats van decimale getallen kunnen ook hexadecimale, octale of binaire getallen gebruikt worden. Dan moet het teken 'F' vervangen worden door bv. X,O of B.
Als de functie ook'don't cares' bevat wordt de beschrijving ingewikkelder. Dan moeten twee van de drie mogelijke waarden aangegeven worden. Dit kan op de volgende manier geschieden:

```
X=F(1(0,1,4)X(7)) of A,B,C;
```

In dit voorbeeld heeft X de waarde 1 bij de combinaties 0,1 en 4, de 'waarde' 'don't care' bij de combinatie 7 en de waarde 0 bij de resterende combinaties.

Als laatste noodzakelijke onderdeel dienen nog de verbindingen tussen de verschillende subsystemen en inputs en outputs van de (sub)systemen beschreven te worden.
Verbindingen tussen subsystemen kunnen beschreven worden als Boolse functie met 1 variabele. Stel P is een output van subsysteem X, Q een input van subsysteem Y en P moet met Q verbonden worden. Dit kan in het omhullende systeem beschreven worden als:

```
Q:=P;
```

3.5 Faciliteiten in DSL2.

In hoofdstuk 2 is reeds gemeld dat in DSL2 de beschrijving zo kort mogelijk wordt gehouden door de invoering van indices, macro instructies en een source statement library. Als verdere faciliteiten van DSL2 kunnen worden genoemd: structuren in registers e.d., expressies aanduiden met een enkel symbool en

insluiten van afzonderlijke (sub)systemen.

Een voorbeeld voor het gebruik van indices is reeds gegeven in programma 2 van hoofdstuk 3.2. Bij de beschrijving van C is gebruik gemaakt van het feit dat door het specificeren van COUNTER als 2 bit teller de naam COUNTER geïndexeerd kan worden om de variabelen van de teller aan te duiden. Hieruit blijkt dat de 0 als laagste index gebruikt wordt terwijl de naam zonder indices de gehele matrix beschrijft.
In REG(0,7) worden de eerste 8 elementen van de matrix REG beschreven.

Het gebruik van indices bleek ook een noodzakelijkheid te zijn voor de toepassing van het GENERATE algorithme (zie hoofdstuk 3.4). Zo kan voorbeeld 3 uit hoofdstuk 3.4 als volgt worden weergegeven:

```
1     SHIFTREG0:              DFF      D=DATA
                                       C=CLOCK;
2     . GENERATE FOR &I=1 TO 3;
3     SHIFTREG&I              DFF      D=SHIFTREG&(&I-1)
                                       C=CLOCK;
4     .                       GEND;
```

Om de beschrijving van de te genereren statements te kunnen onderscheiden van normale statements, wordt de beschrijving voorafgegaan door '.'. Het einde van het GENERATE algorithme wordt aangegeven door 'GEND;'. De genereer statement bevat een variabel symbool, aangeduid met &(naam). Dit symbool wordt in de beschrijving van de te genereren statements gebruikt om de waarde van de variabelen aan te duiden. De naam is noodzakelijk als een dergelijk GENERATE-blok andere GENERATE-blokken bevat.
In bovenstaande beschrijving van het 4-bit schuifregister is het increment van de for loop gelijk aan 1. Algemeen kan worden genoteerd:

```
. GENERATE FOR &(naam)=(expr) STEP (expr) TO (expr);
```

Een andere methode om een aantal statements verkort te beschrijven wordt gegeven door de macro-instructies. Met een macro-instructie wordt in een statement een aantal instructies beschreven. Tijdens het vertaalproces wordt de macro vervangen door

de statements, die beschreven worden door de macro-instructie. De macro-instructies komen overeen met de functie-definitie faciliteit in programmmeertalen. Met macro-instructies kunnen o.a. standaardprocedures en subsystemen worden beschreven. Ook kan de macro worden gebruikt voor het verkort weergeven van veel voorkomende gelijke beschrijvingen.
Om een macro-definitie van de normale beschrijving te kunnen onderscheiden wordt als beginstatement 'MACRO;' gekozen en als eindstatement '.MEND;'
De eerste statement na de 'MACRO' statement beschrijft steeds het prototype. Het prototype is opgebouwd uit vaste en variabele velden. De variabele velden specificeren variabelen. De waarde van deze variabelen komt in de macro-instructie voor. Om de variabele velden van de vaste velden te onderscheiden moeten de variabele velden met '&' beginnen. Het einde van een variabel veld wordt aangegeven door een speciaal teken of blank. Het einde van een vast veld wordt aangegeven door '&'of';'.
De waarde van de variabele velden kan bepaald worden op twee manieren nl.:

a. Door de positie van het variabel veld van het prototype.
b. Door een keywoord, gedefinieerd in het prototype, voor de waarde

Voorbeeld. Beschouw het volgende prototype en macro-instructie:

```
&NAAM:      REG     &LENGTE     BIT;
REG:        REG     16  BIT;
```

Door de positie van de variabele &NAAM wordt diens waarde bepaald. Deze is hier REG. De waarde van &LENGTE is 16.
Beschouw nu het volgende prototype en macro-instructie:

```
&NAAM:      REG     &LENGTH=1;
REG:        REG     LENGTH=16;
```

Hier wordt de waarde van &NAAM weer bepaald door de positie, doch de waarde van &LENGTH wordt nu aangeduid met het keywoord LENGTH.
Zoals gebruikelijk verwerkt de macro processor tekst en wordt aan deze tekst geen verdere betekenis toegekend. Een uitzondering hierop zijn de variabele symbolen en besturingsinstructies. De variabele symbolen worden vervangen door hun waarde, waarbij

de waarde een reeks tekens is. De besturingsinstructies worden gebruikt om (conditioneel) tekst te genereren. Om de besturingsstatements eenvoudig van datastatements te kunnen onderscheiden beginnen besturingsstatements met het teken '.'.

```
IF (bewering) THEN . (naam);
GO TO .(naam);
```

Door deze statements treedt er (conditioneel) een sprong op naar het statement met label '.(naam)'. Dit is een besturingsstatement omdat deze met het teken '.' begint.
De bewering wordt als variabel symbool met een waarde aangegeven bv.:

```
&RESET='YES'
```

Deze bewering is waar als de waarde van de variabele &RESET bestaat uit de reeks 'YES'.

Vaak is het wenselijk om een register, of een andere een-dimensionale matrix, onder te verdelen in subregisters. Zo kan een instructie-register van een rekenmachine worden onderverdeeld in een register voor de operatiecode en registers voor de adressen. Dit kan in DSL2 op twee manieren worden beschreven:

```
INSTRREG:       REG         LENGTH=40;
OPCODE:         EQU         INSTRREG(0,7);
ADDRESS1:       EQU         INSTRREG(8,23);
ADDRESS2:       EQU         INSTRREG(24,39);
```

De andere beschrijvingsmogelijkheid is als volgt:

```
OPCODE:         REG         LENGTH=8;
ADDRESS1:       REG         LENGTH=16;
ADDRESS2:       REG         LENGTH=16;
INTRREG:        EQU         OPCODE,ADDRESS1,ADDRESS2;
```

Met de instructie EQU kan ook aan een symbool de waarde van een expressie toegekend worden, bv.:

```
A: EQU   3+4;
```

De mogelijkheid om vanuit de bibliotheek (sub)systemen in de te verwerken beschrijving op te nemen wordt aangegeven door de instructie:

```
INCLUDE (verzameling)
```

waarbij (verzameling) de verzameling van de gewenste (sub)systemen bevat.

VOOR EEN UITGEBREIDE BESCHRIJVING VAN DSL2 WORDT VERWEZEN NAAR DE APPENDIX VAN LITERATUUROPGAVE 1.

4. ANALYSE VAN SYSTEMEN BESCHREVEN IN DSL2

4.1 Inleiding.

Zoals in hoofdstuk 3.2 is gebleken kan met DSL2 zowel een bestaand systeem als een te ontwerpen systeem worden beschreven. De analyse van een bestaand systeem is bij deze studie niet interessant. Wel van belang is de analyse van een DSL2-beschrijving van een te ontwerpen systeem.
Een DSL2-beschrijving van een te ontwerpen systeem bestaat uit:

a. Procedures
b. Declaraties
c. Commentaar

Een procedure is volgens hoofdstuk 3.2 een beschrijving van een onafhankelijk subsysteem dat via een netwerk gekoppeld is met andere subsystemen. Om de samenwerking te bestuderen tussen het subsysteem beschreven in de procedure en andere subsystemen, wordt het netwerk verdeeld in twee subsystemen: het Control Sysytem en het Data System, respectievelijk afgekort tot CS en DS. Hierbij wordt het CS gelijk gekozen aan het subsysteem beschreven in een procedure. Is dit procedure P dan geldt CS=P, waarbij P wordt gedefinieerd door definitie 3.2.1.

Definitie 4.1.1
Het control system (CS) wordt beschreven in 1 procedure. Dit is het 6-tuple:

$$CS=(S,T,I,O,d,l)$$

met:

S: een eindige niet lege verzameling toestanden
T: initiele toestand
I: een eindige niet lege verzameling inputs
O: een eindige niet lege verzameling outputs
d: $S * I \dashrightarrow S$ een partiele volgende toestand fuctie
l: $S \rightarrow O$ een output functie

Volgens deze definitie is het CS een systeem met toestanden, inputs en outputs. De outputs zijn alleen afhankelijk van de toestanden.

Definitie 4.1.2
Het data system (DS) is een verzameling elementen gedefinieerd door:

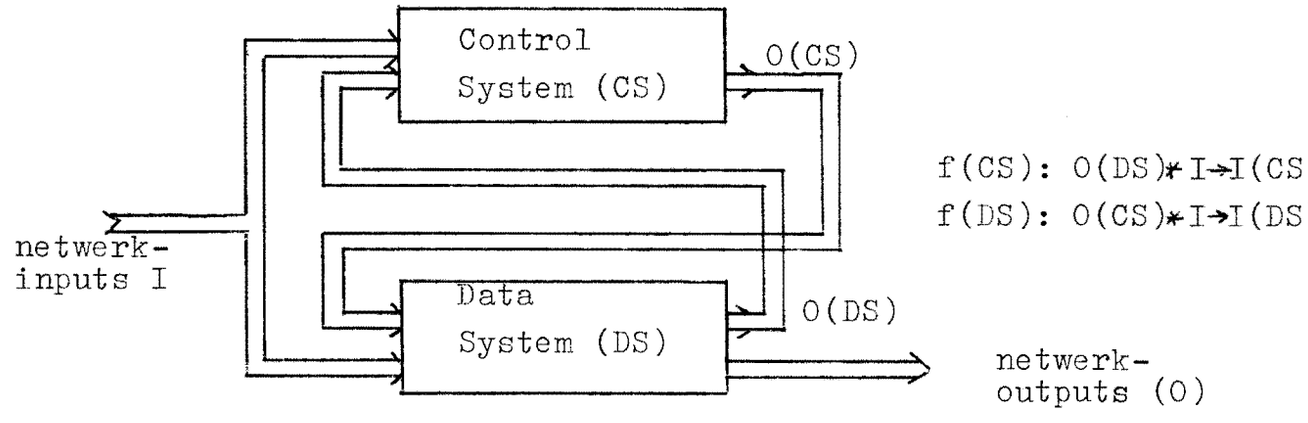
1. Input variabelen x(i) met $1 \leq i \leq m$
2. Toestand variabelen y(j) met $1 \leq j \leq n$
3. Overgang functies:
   $Y(j): \{(y(1),....y(n),x(1),...x(m))\} \rightarrow \{0,1\}$ met $1 \leq j \leq n$
4. Outputfuncties:
   $z(k): \{(y(1),....y(n),x(1),...x(m))\} \rightarrow \{0,1\}$ met $1 \leq k \leq r$

Het CS en DS worden opgenomen in het volgende netwerk:

Definitie 4.1.3
Het netwerk met het CS en DS bestaat uit:

1. De subsystemen CS en DS
2. Een eindig aantal inputs I
3. Een eindig aantal outputs O
4. $f(CS): O(DS) * I \rightarrow I(CS)$
   $f(DS): O(CS) * I \rightarrow I(DS)$
5. $1: O(DS) * I \rightarrow O$

Figuur 4.1.1 geeft het netwerk gedefinieerd door definitie 4.1.3.



Figuur 4.1.1: Het netwerk bestaande uit het control system en het data system. (zie definitie 4.1.3)

Volgens deze definities is de output van het CS alleen afhankelijk

van de toestanden. De inputs van het CS zijn afkomstig van de inputs van het netwerk en van het DS (f(CS)). Alle outputs van het CS gaan naar het DS.
Het DS is gedefinieerd als een verzameling elementen. Dit zijn tenminste alle elementen, met uitzondering van de inputs van het netwerk, die in de procedure beschreven worden. Dat volgt uit de verbindingen in het netwerk. De inputs van het DS zijn afkomstig van de inputs van het netwerk en de outputs van het CS.(f(DS)). Het DS verzorgt alle externe outputs. Het DS kan dus opgevat worden als het gehele beschreven systeem met uitzondering van de procedure die het CS beschrijft en de netwerkinputs.
De samenwerking tussen het CS en DS komt tot uiting in de functies f(CS) en f(DS) uit definitie 4.1.3. Het DS bestaat uit een verzameling elementen die de elementaire bewerkingen uitvoeren en door de verbinding met het CS de uitvoering van het CS conditioneert. Het CS neemt informatie op uit het DS en de netwerkinputs en bepaalt hieruit nieuwe informatie. Een deel van deze informatie blijft in het CS(als de volgende toestand) en een deel gaat naar het DS (output van het CS). De verbinding van het CS naar het DS stuurt de elementaire bewerkingen in het subsysteem DS.

Voorbeeld:
Beschouw de subsystemen M' en M'',
M'=(S',T',I',d',l',O') M''=(S'',T'',I'',O'',d'',l'')
waarbij M' het control system en M'' het data system is.
M' en M'' worden door de volgende tabellen beschreven:

	a	b	c	d	
1	1	2	-	-	w
2	-	2	3	-	x
3	-	-	3	4	y
4	1	-	-	4	z

M'

	a	b	c	d	e	f	g	h
1	1	2	1	1	1	2	1	1
2	2	2	2	1	2	2	2	1

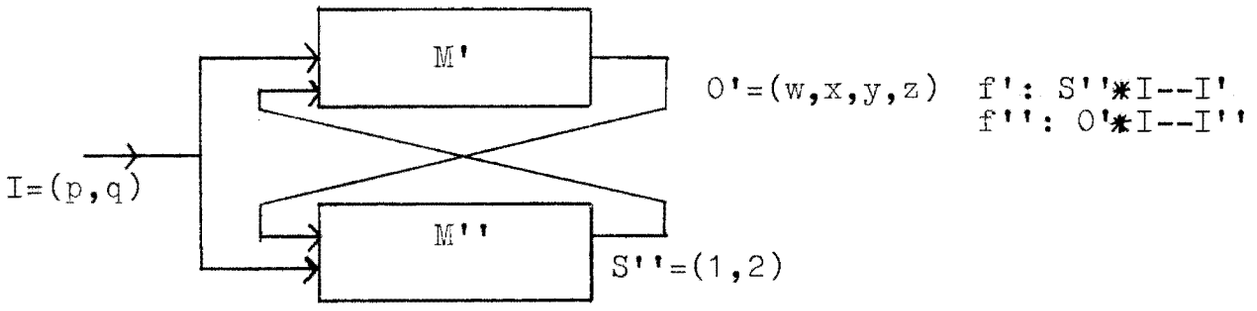
M''

De subsystemen M' en M'' zijn opgenomen in het netwerk met verbindingsfuncties:

f': S''*I--I'
f'': O'*I--I''

De netwerkinputs worden gegeven door I=(p,q), terwijl de net-

werkoutputs niet worden beschouwd.
De subsystemen M', M'' en de verbingsfuncties f' en f'' beschrijven het netwerk M=(S,T,I,d).
Het systeem M is weergegeven in figuur 4.1.2.



Figuur 4.1.2: Systeem M opgebouwd uit de subsystemen M' en M''.

Uit de toestandstabellen van de subsystemen M' en M'' kunnen met de netwerkinputs de verbindingsfuncties f' en f'' worden afgeleid.

f': a=(1,p)
b=(1,q)
c=(2,q)
d=(2,p)

f'': a=(w,p)
b=(x,p)
c=(y,p)
d=(z,p)
e=(w,q)
f=(x,q)
g=(y,q)
h=(z,q)

Stel dat M' in toestand 3 is, M'' in toestand 2 en dat netwerkinput p aanwezig is. Dit levert voor M' de input d=(2,p). M' gaat met input d van toestand 3 naar toestand 4 en heeft hierbij output y. De input van M'' is dan c=(y,p). Bij input c gaat M'' van toestand 2 naar toestand 2. Het systeem M gaat dus bij input p van M' in 3 en M'' in 2 naar M' in 4 en M'' in 2. Op analoge manier kunnen alle andere combinaties worden afgeleid van netwerkinput, toestand van systeem M' en toestand van M''. Dit levert de volgende tabel voor systeem M:

	p	q
11	11	21
12	-	-
21	-	22
22	-	32
31	-	-
32	42	32
41	11	-
42	41	-

4.2 Bepaling van de elementen van het Control System

Uit de beschrijving van de procedure dienen de elementen van het CS (d.w.z. S,T,I,O,d,l) bepaald te worden.

Bepaling van S

Een procedure is opgebouwd uit een aantal statements. Bij het uitvoeren van een procedure wordt steeds een statement tegelijk verwerkt. De instructies beschreven in dit statement, worden parallel uitgevoerd. In hoofdstuk 3.3 is reeds besproken welke soort instructies er in DSL2 kunnen voorkomen. Als een van de instructies in het statement een actie beschrijft, zal dit worden gerealiseerd door de bouwsteen waarin de actie plaatsvindt. Dit geschiedt door de betreffende input van de bouwsteen 1 of 0 te maken. Omdat een instructie welke een actie beschrijft enige tijd vergt om uitgevoerd te worden dient elk statement in het CS, waarin een actie wordt beschreven te corresponderen met een toestand in het CS.

Bepaling van T

Als S(1) het eerste statement van de procedure is die een actie beschrijft dan is s(1) de initiele toestand van het CS zodat dan T=s(1). Statement S(1) wordt verwerkt als het CS zich in toestand s(1), de initiele toestand bevindt.

Bepaling van I

De verwerking van de procedure hangt af van de toestand van de elementen van het DS en de netwerkinputs. (Zie voorbeeld hoofdstuk 4.1). Dus als alle elementen van het DS en de netwerkinputs van het CS beschouwd worden, zijn er voldoende inputs aanwezig om de volgende toestand en output van het CS te bepalen.

Bepaling van O

De outputs van het CS verzorgen de informatieoverdracht van het CS naar het DS (zie figuur 4.1.2,hoofdstuk 4.1). Derhalve dient iedere instructie in de procedure die een actie op een element uit het DS beschrijft, overeen te komen met een output variabele van het Control System als veronder-

steld wordt dat alle instructies onafhankelijk van elkaar zijn. In het algemeen zullen niet alle instructies onafhankelijk van elkaar zijn. Dit betekent dat niet alle mogelijke output toestanden beschreven worden.

Bepaling van d

De volgende toestand functie d geeft bij een toestand s van het CS en een input x de volgende toestand. Wanneér een toestand overgaat in een volgende toestand is afhankelijk van het systeem.

In synchrone systemen verandert de toestand op commando van een kloksignaal. In asynchrone systemen verandert de toestand spontaan. Bij zowel synchrone als asynchrone procedures mag een toestand pas overgaan naar een volgende toestand als de instructies, beschreven in het statement dat correspondeert met de momentane toestand, uitgevoerd zijn. Hieruit volgt dat het niet nodig is om onderscheid te maken tussen synchrone en asynchrone procedures.

Tengevolge van de uitvoering van de instructies krijgen de toestanden van de elementen van het DS bepaalde waarden. Een instructie kan dus als uitgevoerd beschouwd worden als de toestanden van de elementen, die door de instructie bewerkt worden, de waarde hebben die zij tengevolge van de uitvoering van de instructie zouden krijgen. Indien niet bepaald kan worden wanneer een instructie is uitgevoerd, kan gesteld worden dat de uitvoering direct na de start gereed is.
Als een statement is uitgevoerd, wordt de volgende toestand bepaald door het volgende statement in de procedure, tenzij door een besturingsinstructie naar een andere statement wordt verwezen.
De volgende toestand functie is een partiële afbeelding van S*I op S. Dit wordt veroorzaakt doordat bij een toestand van het CS niet alle inputs mogelijk zijn, zodat niet bij alle combinaties van toestand en input een volgende toestand aangegeven kan worden. De niet gedefinieerde waarden worden door de specificatie van de partiële functie aanvaard.
Naast de overgang naar een volgende toestand kan de procedure ook nog inputs beschrijven waarbij de toestand gelijk blijft.

Bepaling van l

De output functie l geeft de output 0 als functie van de toestanden van het CS. De output is onafhankelijk van de input van het CS (zie definitie 4.1.1).
Zoals bij de bepaling van 0 bleek, verzorgt de outputvariabele o(j) de uitvoering van instructie I(j). De instructie I(j) dient uitgevoerd te worden als het statement dat I(j) bevat, verwerkt wordt. Dan moet dus gelden: o(j)=1.
Het is echter niet altijd nodig dat instructie I(j) niet uitgevoerd wordt als alle andere statements verwerkt worden. Bv. een SET-instructie mag altijd uitgevoerd worden als de toestand van de elementen, die door de SET-instructie de waarde 1 zouden krijgen reeds de waarde 1 hebben.
Een uitzondering hierop vormt een strijdige instructie. Dit is een instructie, die een of meerdere dezelfde elementen verwerkt doch hun toestand een andere waarde kan geven.
Als een instructie bij het verwerken van andere statements uitgevoerd mag worden kunnen er minder zware eisen aan de outputs van het CS gesteld worden zodat eenvoudiger realisaties mogelijk zijn. We kunnen daarom beter aangeven of bij een bepaalde statement de uitvoering van een instructie vereist, verboden of vereist noch verboden is. De eisen voor de uitvoering van een instructie worden executie voorwaarden genoemd.
De waarden van de executie voorwaarden zijn:

a. 1 bij die toestanden van het CS waarbij de actie uitgevoerd moet worden
b. 0 bij die toestanden van het CS waarbij een strijdige instructie uitgevoerd moet worden
c. X bij die toestanden van het CS waarbij de toestanden van de elementen die door de instructie bewerkt worden reeds de waarde hebben die zij t.g.v. de uitvoering van de instructie zouden krijgen.
d. 0 bij alle andere toestanden van het CS.

De output functie l bestaat dus uit de verzameling van de executie voorwaarden van de outputs, als functie van de toestanden.

Voorbeeld 1

```
1  PROCEDURE;
2        STATE OF SELECT IS 0;
3        WAIT FOR SELECT IS 1;
4        STATE OF RDY IS 0;
5        WAIT FOR RDY IS 1;
6        LOAD REG;
7        WAIT FOR RDY IS 0;
8        IF PARITY=1 THEN SET CHECK;
9        WAIT FOR SELECT IS 0;
10 END;
```

De procedure beschrijft het laden van een register en de daarbij behorende parity controle. De procedure begint met te wachten tot het register geselecteerd wordt. Daarna wordt gewacht tot de aangeboden data geldig is (is geldig als RDY=1). Dan wordt het register geladen. Nadat RDY afgevallen is wordt de pariteit gecontroleerd en indien fout CHECK geset. Tenslotte wordt gewacht tot SELECT weer afgevallen is.

In de procedure worden in de statements 3,5,6,7,8 en 9 acties beschreven (WAIT,LOAD,SET). Het CS kent dus 6 toestanden welke overeenkomen met de genoemde statements.
De inputs van het CS worden gerepresenteerd door de mogelijke toestanden van de elementen van het DS en de netwerkinputs.
In de procedure worden de volgende elementen genoemd: SELECT, RDY, REG, PARITY, CHECK. De toestanden van REG zijn niet significant zodat deze weggelaten kunnen worden. Met de 4 resterende elementen zijn 2^4 inputs mogelijk welke in de volgende tabel worden weergegeven.

element	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
SELECT	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
RDY	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
PARITY	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
CHECK	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Omdat de instructies in statements 6 en 8 een aktie beschrijven moeten deze overeenkomen met een outputvariabele. Het CS heeft dus 2 outputvariabelen. Deze besturen de uitvoering van de instructies beschreven in de statements 6 en 8. De WAIT-instructie in statements 3,5,7 en 9 levert geen output variabele.

Verder kan uit deze procedure de volgende tabel voor de volgende toestand functie en de executie voorwaarden worden afgeleid.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	6	8
3	3	3	3	3	3	3	3	3	5	5	5	5	-	-	-	-	0	0
5	-	-	-	-	-	-	-	-	5	5	5	5	6	6	6	6	0	0
6	-	-	-	-	-	-	-	-	-	-	-	-	7	7	7	7	1	0
7	-	-	-	-	-	-	-	-	9	9	8	8	7	7	7	7	X	0
8	-	-	-	-	-	-	-	-	-	-	8	9	-	-	-	-	0	1
9	3	3	-	3	-	-	-	-	9	9	-	9	-	-	-	-	0	0
	Volgende toestand functie																executie voorwaarden	

De bepaling van de volgende toestand functie d en de executie voorwaarden zal worden besproken in hoofdstuk 4.3.

Het voorbeeld toont aan dat de volgende toestand tabel zeer groot wordt bij een groot aantal inputs. Ook blijkt er bij de meeste combinaties van toestand en input geen volgende toestand tabel te noteren door het beschrijven van de toestand sequenties. (zie figuur 3.1.5 en figuur 3.1.6 van hoofdstuk 3.1). Tevens is het dan mogelijk om de inputs te beschrijven met de input variabelen.
Dit toegepast op het voorbeeld levert de volgende tabel:

d(s,x)	s	SELECT	RDY	PARITY	CHECK	6	8
3	3	0	X	X	X	0	0
3	9	0	0	0	X	0	0
3	9	0	0	1	1	0	0
5	3	1	0	X	X	0	0
5	5	1	0	X	X	0	0
6	5	1	1	X	X	1	0
7	6	1	1	X	X	X	0
7	7	1	1	X	X	X	0
8	7	1	0	1	X	0	1
8	8	1	0	1	0	0	1
9	7	1	0	0	X	0	0
9	8	1	0	1	1	0	X
9	9	1	0	0	X	0	0
9	9	1	0	1	1	0	X
toestand	vorige toestand	toestand tijdens executie				output	

Tabel 4.2.1: Beschrijving van de volgende toestand tabel door toestand sequenties en input variabelen.

In de tabel is de toestand van een input variabele als X genoteerd als deze 1 of 0 kan zijn.

Behalve een kortere notatie van de volgende toestand tabel heeft het beschrijven van de toestand sequenties nog de volgende voordelen:

1- De beschrijving van de toestand sequenties sluit beter aan bij de werking van de procedure omdat de inputs kunnen worden gegeven door de input variabelen. Dit zal nader blijken bij de ontwerpsimulatie welke in hoofdstuk 4.3 aan de orde komt.
2- De beschrijving van de toestand sequenties kan worden gebruikt om de logische vergelijkingen van het systeem te bepalen. (zie voorbeeld JK-flipflop, figuur 3.1.4/3.1.7). Dit zal worden besproken in hoofdstuk 6.

4.3 Bepaling van de volgende toestand functie en de executie voorwaarden van het CS door simulatie.

Om na te gaan bij welke toestanden van de input variabelen en welke toestand van het CS het DS in een bepaalde toestand komt, wordt het systeem gesimuleerd. De simulatie levert dan de volgende toestand functie.
Bij de simulatie van de procedure zullen de inputs van het CS gerepresenteerd worden door de toestanden van de input variabelen. Als e(1),.....e(m) de verzameling input variabelen is dan representeerd het m-tuple a(1),.....a(m) de mogelijke toestanden als voor a(k) de toestand van de variabele e(k) wordt ingevuld. Deze toestand is 0,1 of X.
Om de volgende toestand functie te bepalen kan de procedure onder alle mogelijke omstandigheden worden gesimuleerd. Het is echter niet nodig om de procedure onder alle mogelijke omstandigheden te simuleren als de instructies op zodanige wijze worden gesimuleerd dat deze simulatie representatief is voor de simulatie onder alle mogelijke omstandigheden. Bij een representatieve simulatie behoeft slechts de waarde van het m-tuple, die de toestanden van de te bewerken elementen van het DS representeren, gewijzigd te worden in een waarde die de uitvoering onder alle mogelijke omstandigheden representeren.

De representatieve simulatie kan het best worden geïllustreerd met enkele voorbeelden:

Voorbeeld 1

Beschouw de instructie:

SET (set)

(set) specificeert een aantal geheugenelementen. De uitvoering van deze instructie is onafhankelijk van een voorwaarde, dus bij een representatieve simulatie kan de toestand van de elementen, beschreven door (set) op 1 gezet worden.

Voorbeeld 2

Beschouw de instructie:

COUNT(set)

(set) specificeert een teller. Bij de uitvoering van de instructie wordt de inhoud van de teller verhoogd (of verminderd) met 1. De toestand na uitvoering is dus afhankelijk van de vorige toestand. Dus wordt bij een representatieve simulatie de toestand van de elementen, aangegeven door (set), op X gezet.

Voorbeeld 3

Beschouw de instructie:

IF (bewering) THEN (instr1) ELSE (instr2)

Bij een representatieve simulatie moet deze instructie tweemaal worden gesimuleerd. Bij de eerste simulatie wordt verondersteld dat (bewering) waar is en wordt (instr1) verwerkt. Bij de tweede simulatie wordt gesteld dat (bewering) niet waar is en wordt (instr2) verwerkt.

Door een representatieve simulatie wordt de volgende toestand van het CS met de daarbij behorende inputs bepaald. Dit wordt genoteerd als:

(r,A)=SIM(s,B)

met s= toestand van het CS

B=deelverzameling van de inputs waarmee het CS in s komt

A=deelverzameling van de inputs waarmee het CS in r komt

r=d(s,A) de volgende toestand van het CS

Stel dat de toestanden van de input variabelen in toestand s gelijk zijn aan $a_s(1),....a_s(i),....a_s(m)$ en dat de instructie

input variabele e(i) bewerkt. Dan zal de toestand van e(i) na uitvoering van de instructie gelijk zijn aan:

$a_r(i)$ =1 als de toestand van e(i) na uitvoering van de instructie altijd 1 is

=0 als de toestand van e(i) na uitvoering van de instructie altijd 0 is

=X in andere gevallen.

Hierbij is $a_r(i)$ de toestand van de input variabele e(i) waarbij het systeem overgaat van toestand s naar r. Voor de overige input variabelen zal bij de overgang van toestand s naar r gelden:

$a_r(j)=a_s(j)$ voor $1 \leq r \leq (i-1)$

$(i+1) \leq r \leq m$

Uitgaande van $a_r(1)/a_r(m)$ kunnen weer de toestanden van de input variabelen worden afgeleid waarbij toestand r overgaat in een volgende toestand enz. enz.. De representatieve simulatie is dus een recursieve bewerking waarbij het volgende resultaat afhankelijk is van het voorafgaande resultaat. Voor de recursieve bewerking zal dus een initiele waarde moeten worden gedefinieerd en een voorwaarde, waarbij de bewerking kan stoppen.
Aan het begin van de simulatie zal van geen enkele input variabele bekend zijn, in welke toestand deze zich bevindt. We kunnen daarom de toestand van de input variabele dan voorstellen door X. De eerste toestand zal worden gegeven door het eerste statement dat een actie beschrijft. Uit de beschrijving van de procedure volgt verder de volgende toestand bij een bepaalde toestand.
De simulatie kan worden beëindigd als de verzameling inputs waarmee een instructie instructie gesimuleerd zou worden een deelverzameling is van de vereniging van deelverzamelingen waarmee de instructie reeds gesimuleerd is. Dan levert de simulatie immers geen nieuwe toestanden van input variabelen.

De simulatie kan nu als volgt worden uitgevoerd:

1. De statements in de procedure worden verwerkt in de volgorde zoals aangegeven in de procedure.
2. De instructies in de statements worden op representatieve gesimuleerd
3. De verzameling inputs waarmee het eerste statement gesimuleerd wordt bevat alle toestanden

4. Als een statement verwerkt is wordt de verkregen verzameling inputs gebruikt bij de verwerking van alle volgende statements
5. Als een statement meerdere volgende statements heeft wordt de simulatie met een willekeurig volgend statement voortgezet. De verzameling inputs waarmee de volgende statements verwerkt moeten worden wordt bewaard zodat deze statements later verwerkt kunnen worden.
6. Een statement wordt alleen dan verwerkt als de verzameling inputs waarmee het statement verwerkt moet worden niet omvat wordt door de vereniging van verzamelingen waarmee het statement reeds gesimuleerd is.
7. Als de simulatie volgens stap 6 afgebroken wordt, wordt deze voortgezet met de verwerking van de statements die bij stap 5 bewaard zijn. Als dergelijke statements niet aanwezig zijn is de simulatie gereed.

De simulatie sluit dus direct aan bij de notatie van een volgende toestand tabel volgens toestand sequenties. In deze tabel werden de inputs ook beschreven met input variabelen.

Met behulp van de volgende toestand functie kan er een executie voorwaarde voor iedere actie worden gevonden. De waarde van de executie voorwaarde is:

1 bij die toestanden van het CS waarbij de actie uitgevoerd moet worden,
0 bij die toestanden van het CS waarbij een strijdige instructie uitgevoerd moet worden,
X bij die toestanden van het CS waarbij de toestanden van de elementen die door de instructie bewerkt worden reeds de waarde hebben die zij t.g.v. de uitvoering van de instructie zouden krijgen,
0 bij alle andere toestanden van het CS.

Het is dus mogelijk om door representatieve simulatie, de volgende-toestand tabel (en de outputtabel) uit de beschrijving van de procedure af te leiden. De afleiding kan in het DSL2 systeem geschieden door een computerprogramma, dat hiervoor is geschreven.

Voorbeeld

Als voorbeeld zal de volgende toestand functie en de executie voorwaarden voor de procedure uit hoofdstuk 4.2 worden behandeld.

```
 1   PROCEDURE;
 2         STATE OF SELECT IS 0;
 3         WAIT FOR SELECT IS 1;
 4         STATE OF RDY IS 0;
 5         WAIT FOR RDY IS 1;
 6         LOAD REG;
 7         WAIT FOR RDY IS 0;
 8         IF PARITY=1 THEN SET CHECK;
 9         WAIT FOR SELECT IS 0;
10   END;
```

We beschouwen de input variabelen steeds in de volgorde:

SELECT,RDY,PARITY,CHECK.

Verder worden de resultaten weergegeven in de volgorde:

toestand	vorige toestand	toestand van de input variabelen tijdens de executie

Deze notatie is analoog aan die uit tabel 4.2.1.

Voordat de procedure wordt uitgevoerd is er van geen enkele input variabele bekend in welke toestand deze zich bevindt. We kunnen daarom de toestanden van de input variabelen voorstellen door

X,X,X,X

waarbij X er op duidt dat de toestand van de input variabele 0 of 1 kan zijn.
In statement 2 van de procedure wordt de toestand van SELECT 0 en het systeem bevindt zich in de initiele toestand 3. Het systeem zal in toestand 3 blijven zolang SELECT=0. Er geldt dus:

3 3 0,X,X,X

Als SELECT=1 zal het systeem overgaan naar een volgende toestand. Door de beschrijvende instructie in statement 4 zal hierbij RDY de waarde 1 krijgen. De overgang van toestand 3 naar toestand 5 zal dus optreden als de toestanden van de input variabelen respectievelijk gelijk zijn aan 1,0,X,X. Dit geeft:

5 3 1,0,X,X

Van de eerder verkregen inputverzameling (0,X,X,X) zijn dus alleen de toestanden van die input variabelen veranderd, welke door de instructies werden bewerkt.De nu verkregen input verzameling (1,0,X,X) wordt weer gebruikt bij de verwerking van alle volgende statements.

Het systeem zal in toestand 5 blijven voor 1,0,X,X, zodat geldt

5 5 1,0,X,X

Als RDY=1 gaat het systeem over van toestand 5 naar toestand 6.

6 5 1,1,X,X

Voor de overgang van 6 naar 7 zullen de toestanden van de input variabelen niet veranderen zodat dezelfde input verzameling wordt gevonden als bij de overgang van 5 naar 6. De input variabele REG zal bij de representatieve simulatie in toestand 6 niet van toestand veranderen d.w.z. de toestand van REG was X en blijft X. De toestanden van REG zijn dus niet significant zodat ze weggelaten kunnen worden.
Het systeem zal zijnde in toestand 7, in toestand 7 blijven voor 1,1,X,X.

7 6 1,1,X,X
7 7 1,1,X,X

Statement 7 heeft meerdere volgende statements, nl. 8 of 9. Dit betekent dat de inputverzameling 1,1,X,X moet worden onthouden.
Toestand 7 gaat over naar toestand 9 als RDY=0 en PARITY=0. Dit geeft: 9 7 1,0,0,X
Het systeem zal bij dezelfde inputverzameling in toestand 9 blijven: 9 9 1,0,0,X
Voor SELECT=1 treedt een overgang op van toestand 9 naar toestand 3: 3 9 0,0,0,X
Deze inputs vormen een deelverzameling van de inputs waarmee toestand 3 reeds gesimuleerd is. (0,X,X,X). Dit houdt in dat de simulatie op dit punt kan worden afgebroken en dat kan worden voortgezet met de verwerking van statement 8.
De verzameling inputs waarvan weer wordt uitgegaan is 1,1,X,X.
Toestand 7 gaat over naar toestand 8 als RDY=0 en PARITY=1 zodat 8 7 1,0,1,X
Zolang CHECK de waarde 0 heeft zal het systeem in toestand 8 blijven: 8 8 1,0,1,0
Het systeem zal van toestand 8 naar toestand 9 gaan bij CHECK=1.

9 8 1,0,1,1

Het systeem gaat van toestand 9 naar toestand 3 als SELECT=0,

3 9 0,0,1,1

en zal in 9 blijven als SELECT=1 blijft.

9 9 1,0,1,1

Omdat 0,0,1,1 een deelverzameling is van de vereniging van deelverzamelingen waarmee statement 3 reeds verwerkt is (0,X,X,X en 0,0,0,X) kan de simulatie als gereed worden beschouwd. Het resultaat van de simulatie is de tabel welke reeds gevonden is in hoofdstuk 4.2 (tabel 4.2.1).

Voor het afleiden van de executie voorwaarden kan nu gebruik worden gemaakt van de toestand tabel beschreven door toestand sequenties en input variabelen. De waarden van de executie voorwaarden kan bepaald worden zoals besproken op blz. 47.

De executie voorwaarde o(j) voor instructie I(j) is steeds 1 omdat o(j) de uitvoering van I(j) verzorgt. Dus de executie voorwaarde is 1 voor toestand 6, output 6 en toestand 8, output 8.
De waarden van de input variabelen zijn bij de overgang van toestand 6 naar 7 en 7 naar 7 gelijk aan de waarden van de input variabelen bij de overgang van toestand 5 naar 6.
De executie voorwaarde van output 6 is bij de overgangen 6,7 en 7,7 dus gelijk aan X. Bij de resterende toestandsovergangen is de executie voorwaarde voor output 6 gelijk aan 0. Voor output 8 zal de executie voorwaarde X kunnen zijn als CHECK de waarde 1 heeft. Dit geldt echter niet voor de overgang van toestand 9 naar toestand 3 omdat in toestand 3 de waarde van CHECK niet bekend is. Voor deze overgang is de executie voorwaarde voor output 8 0. Bij alle andere toestandsovergangen is de executie voorwaarde voor output 8 ook 0.

Voorbeeld 2

```
1  SYSTEM;
2.   READ:      PROCEDURE;
3               STATE OF SELECT IS 0;
4               RESET BITTELLER, DATARDY;
5               STATE OF BIT9 = 0;
6               WAIT FOR SELECT=1;
7               STATE OF RDY = 0;
8         SHIFT: WAIT FOR RDY = 1;
9               SHIFT SHIFTREG-COUNT BITTELLER;
10              WAIT FOR RDY = 0;        SHIFT
11              IF BIT9 = 0 THEN GO TO SHIFT;
12              IF PARITY = 0 THEN SET DATARDY;
13              WAIT FOR DATARDY = 0;
14              IF SELECT = 1 THEN WAIT FOR SELECT = 0;
15   END;
```

```
16      SHIFTREG:       SR LENGTH = 9 D-INPUT = DATA IN;
17      BITTELLER:      COUNTER LENGTH = 4 BACKWARD RESET = 9;
18      BIT9:           = F(0) OF BITTELLER;
19      DATARDY:        FF;
20      PARITY:         = EVEN PARITY OF SHIFTREG;
21      SELECT,
        RDY,
        DATA IN:        INPUT;
22      DATARDY,
        SHIFTREG:       OUTPUT;
23      END;
```

Deze procedure beschrijft het schuiven van informatie in een schuifregister. Er wordt verondersteld dat dit schuifregister via een bussysteem met andere apparatuur verbonden is. Het schuifregister mag informatie opnemen als het geselecteerd wordt. Dit wordt aangegeven door het signaal SELECT. De informatie op de bus is geldig als RDY 1 is, dus dan kan de informatie in het schuifregister geschoven worden. Verder dient het schuifregister een bitteller bij te houden om de operatie te kunnen beeindigen. Tenslotte wordt een procedure gestart om de informatie te verwerken.
Uit de procedure zijn de volgende tabellen af te leiden:

d(s,x)	s	SE-LECT	RDY	BIT9	PA-RITY	DA-TARDY	4	9	12
4	4	0	X	X	X	X	1	0	0
4	13	0	0	1	X	0	1	0	0
4	14	0	0	1	X	0	1	0	0
6	4	0	X	0	X	0	X	0	0
6	6	0	X	0	X	0	X	0	0
8	6	1	0	0	X	0	0	0	0
8	8	1	0	0	X	0	0	0	0
8	10	1	0	0	X	0	0	0	0
9	8	1	1	0	X	0	0	1	0
10	9	1	1	0	X	0	0	X	0
10	10	1	1	0	X	0	0	X	0
12	10	1	0	1	0	0	0	0	1
12	12	1	0	1	0	0	0	0	1
13	10	1	0	1	1	0	0	0	0
13	12	1	0	1	0	1	0	0	0
13	13	1	0	1	0	1	0	0	0
14	13	1	0	1	X	0	0	0	0
14	14	1	0	1	X	0	0	0	0
toe-stand	vori-ge toe-stand	toestand tijdens executie					output		

Tabel 4.2.2: Volgende toestand tabel en output tabel van voorbeeld 2.

5. MATHEMATISCHE SYNTHESE VAN HET CONTROL SYSTEM.

Om het ontwerpen van een digitaal systeem te automatiseren, zal er een synthese methode moeten worden gedefinieerd welke geschikt is voor de verwerking door een computer. Zoals uit hoofdstuk 4 is gebleken, wordt het Control System beschreven door de volgende toestand- en output tabel. De tabellen bevatten de wezenlijke kenmerken van het te ontwerpen systeem.

Door Hartmanis en Stearns (29) zijn een aantal algebraïsche technieken ontwikkeld om een digitaal systeem te realiseren door het opsplitsen van de digitale machine in component machines. De theorie geeft een aantal structurele realisaties van de machine zonder dat wordt ingegaan op de fysische realisatie van het digitaal systeem. De structurele realisaties worden gevonden aan de hand van een 'mathematisch' optimaliseringscriterium en het blijkt, dat in het algemeen goede c.q. gunstige realisaties ontstaan. De structurele realisaties volgen uit de gekozen toestand toewijzingen.

Voor de mathematische synthese van het Control System wordt uitgegaan van bovenstaande theorie.
Omdat de volgende toestand functie van het Control System gedefinieerd is als een partiele functie en de output tabel 'don't cares' vertoont, wordt het begrip 'Extended Partition' ingevoerd. Dit begrip is synoniem aan het begrip 'Set System' uit hoofdstuk 5 van Hartmanis en Stearns (29). Verder wordt de theorie uitgebreid door het invoeren van het begrip 'Triplet'. Het 'Triplet' wordt gevormd door twee extended partitions P en Q op S en een extended partition E op I. Hierbij is S de verzameling toestanden van het CS en I de verzameling inputs van het CS. In dit hoofdstuk zal worden volstaan met het vermelden van de definities en theorema's welke de mathematische grondslag vormen voor de algebraïsche synthese van het CS. Voor de formele bewijzen van de hiervoor noodzakelijke theorema's zij verwezen naar literatuuropgave 1.

Een verzameling wordt op twee manieren beschreven. Een verzameling wordt beschreven door alle elementen op te noemen of door de eigenschappen van de elementen te specificeren. Dit wordt genoteerd als:

$S = \{ s(1), s(2), \ldots s(n) \}$ noemen van alle elementen

$S = \{ s \mid s \text{ heeft eigenschap } X \}$ beschrijving van de eigenschappen

Een element s van een verzameling S wordt aangeduid met:

s in S

Als s in S impliceert dat s in T dan is S een deelverzameling van T. Dit wordt genoteerd als:

$S \subseteq T$

Twee verzamelingen S en T zijn gelijk dan en slechts dan als:

$S \subseteq T$ en $T \subseteq S$

geschreven als: $S = T$

De doorsnijding van 2 verzamelingen S en T is de verzameling:

$S \cdot T = \{ s \mid s \text{ in } S \text{ en } s \text{ in } T \}$

De vereniging van 2 verzamelingen s en T is de verzameling:

$S + T = \{ s \mid s \text{ in } S \text{ of } s \text{ in } T \}$

Het verschil van 2 verzamelingen S en T is de verzameling:

$S - T = \{ s \mid s \text{ in } S \text{ en niet } s \text{ in } T \}$

Stel S en T zijn niet lege verzamelingen. De functie f:

$f: S \rightarrow T$

definieert bij iedere s in S een t in T. Dit wordt geschreven als:

$f(s) = t$

De functie f wordt ook afbeelding van S op T genoemd. Als:

$T = \{ t \mid t = f(s) \text{ voor } s \text{ in } S \}$

dan is f een 'onto' functie. De functie is 'one-to-one' als:

$s(1) \neq s(2)$ impliceert dat $f(s(1)) \neq f(s(2))$

De functie f is een partiele functie of partiele afbeelding als f gedefinieerd is op een deelverzameling van S.

Zoals reeds is besproken kan een digitaal systeem worden gerepresenteerd door het 6-tuple M:

$$M=(S,T,I,O,d,l)$$

met:

S: een eindige niet lege verzameling toestanden
T: een eindige niet lege verzameling initiele toestanden
I: een eindige niet lege verzameling inputs
O: een eindige niet lege verzameling outputs
d: $S * I \rightarrow S$ een volgende toestand functie
l: $S * I \rightarrow O$ een output functie

Bij een systeem M gaat toestand s over in toestand t bij input x dan en slechts dan als:

$$t = d(s,x)$$

Voor een deelverzameling B van S is:

$$d(B,x) = \{ s \mid s=d(t,x) \text{ en } t \text{ in } B\}$$

Nu gaat de deelverzameling B over in de verzameling B' bij input x dan en slechts dan als:

$$d(B,x) \leqq B'$$

Voor een deelverzameling B van I is:

$$d(s,B) = \{t \mid t=d(s,x) \text{ en } x \text{ in } B\}$$

Nu gaat toestand s over in de verzameling S' bij een input x in B dan en slechts dan als:

$$d(s,B) \leqq S'$$

Een variabele in een systeem kent slechts 2 toestanden, gerepresenteerd door:

a. 0
b. 1
c. X, toestand is 0 of 1, doch welke is onbekend.

Een significante toestand is een toestand die 0 of 1 is.

De toestanden van een systeem met n elementen worden gerepresenteerd door het n-tuple:

$$(e(1),...e(i),...e(n))$$

Hierbij geeft e(i) de toestand van element i aan.
e(i) is:

1 als toestand van element i is 1
0 als toestand van element i is 0
X als toestand van element i is 0 of 1.

Definitie 5.1

Een aantal deelverzamelingen:

$Q = \{B(i)\}$ voor i = 1,2,...n

van S is een extended partition, afgekort tot EP, op S dan en slechts dan als:

1. $\pm B(i) = S$
2. $B(i) \leq B(j)$ impliceert i=j

De tweede voorwaarde zorgt ervoor dat een EP een minimaal aantal elementen heeft.

De elementen van Q worden blokken genoemd. Een blok van Q wordt aangeduid met:

B(Q)

Een blok van Q dat s (s in S) bevat wordt aangeduid met:

B(Q,s)

De elementen s en t (beide in S) zijn equivalent onder de EP Q op S dan en slechts dan als:

B(Q,s)=B(Q,t)

d.w.z. als s en t beiden in hetzelfde blok van Q voorkomen. Dit wordt geschreven als:

$s \equiv t(Q)$

Voorbeeld. 1

De EP $Q = \{1,2,3;1,3,4\}$ bestaat uit twee blokken. Het eerste blok bevat de toestanden 1,2 en 3. Het tweede blok bevat de toestanden 1,3 en 4. De toestanden 1 en 3 komen voor in beide blokken. Blok 1 en blok 2 samen bevatten alle toestanden.(S) Voor het eerste blok geldt: $1 \equiv 2 \equiv 3(Q)$ want toestand 1,2 en 3 komen allen in het eerste blok voor. Analoog voor toestand 1,3 en 4 in het tweede blok.

Definitie 5.2

Stel Q en R zijn EP's op een verzameling S. Dan is:

$$R \leq Q$$

als en slechts als er voor iedere B(R) een B(Q) is zodanig dat

$$B(R) \leq B(Q)$$

Voorbeeld 2

$$\{1,2;2,4;3,4\} \leq \{1,2,3;2,3,4\}$$

Theorema 5.1

Een EP Q op S realiseert een EP R op S dan en slechts dan als:

$$Q \leq R$$

De realisatie van een EP vraagt een onderscheid tussen de verschillende elementen van S. Dit onderscheid hoeft niet nauwkeuriger te zijn dan door de te realiseren EP wordt aangegeven. Iedere EP die kleiner of gelijk is aan de te realiseren EP voldoet aan deze voorwaarde.

Definitie 5.3

Stel $\{S(i)\}$ is een aantal deelverzamelingen van S. Dan is:

$$\text{Max}\{S(i)\} = \{B \leq S \mid B=S(i) \text{ en } B \leq S(j) \text{ impliceert } B=S(j)\}$$

Voorbeeld 3

$$\text{Max}\{1;1,2;2,3;2,3\} = \{1,2;2,3\}$$

Theorema 5.2

Stel Q en R zijn EP's op een verzameling S. Dan is het product Q.R:

$$Q.R = \text{Max}\{B(Q).B(R)\}$$

en de som Q+R:

$$Q+R = \text{Max}\{Q \cup R\}$$

Voorbeeld 4

$$\{1,2,3;3,4,5\}.\{1,2;2,3,4;4,5\} = \{1,2;2,3;3,4;4,5\}$$
$$\{1,2,3;3,4,5\}+\{1,2;2,3,4;4,5\} = \{1,2,3;2,3,4;3,4,5\}$$

Met bovenstaande definities voor vermenigvuldiging en optelling kunnen eenvoudig de volgende beweringen voor de EP's P, Q en R (allen op een verzameling S) bewezen worden:

$Q \geqslant Q.R$

$Q \leqslant Q+R$

Theorema 5.3

Op de verzameling toestanden S van een systeem M kan voor ieder element e(i) een EP E(i) geconstrueerd worden, Symbolisch:

$$E(i) = \text{Max}\{B^1(i), B^0(i)\}$$

met:

$B^1(i) = \{ s \text{ in } S \mid \text{toestand van } e(i) \text{ is 1 of X} \}$

$B^0(i) = \{ s \text{ in } S \mid \text{toestand van } e(i) \text{ is 0 of X} \}$

E(i) is een EP omdat door de Max-operatie het aantal blokken minimaal is en het element i bij iedere s in S een toestand 0 of 1 heeft.

Definitie 5.4

Een significant element is een element waarvan de EP op de verzameling toestanden van het systeem 2 blokken bevat. De EP van een niet significant element bestaat slechts uit 1 blok.

Bij het ontwerp van een systeem zijn slechts significante elementen van belang omdat deze elementen gebruikt kunnen worden om 2 toestanden van het systeem te onderscheiden.

Voorbeeld 5

De EP $E(i) = \{1,2,4;3,4\}$ van element i geeft aan dat uit de toestand van element i bepaald kan worden of het systeem zich in een der toestanden 1,2,4 of in een der toestanden 3,4 bevindt.

Bij het ontwerpen van digitale machines vormt het toekennen van de binaire toestand variabelen om de toestand van de machine aan te geven, een zeer belangrijke stap. In het algemeen zullen verschillende toekenningen resulteren in verschillende logische relaties en de complexiteit zal sterk varieren met de toekenning van de toestand variabelen.

Voorbeeld 6

	inputs 0	1	z
0	3	2	0
1	5	2	0
2	4	1	0
3	1	4	1
4	0	3	0
5	2	3	0

Machine 1

	y_1	y_2	y_3
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1

Toewijzing A

	y_1	y_2	y_3
0	1	1	0
1	1	0	1
2	1	0	0
3	0	0	0
4	0	0	1
5	0	1	0

Toewijzing B

Toewijzing B levert de volgende tabel:

y_1	y_2	y_3	Y_1	Y_2	Y_3	Y_1	Y_2	Y_3	z
1	1	0	0	0	0	1	0	0	0
1	0	1	0	1	0	1	0	0	0
1	0	0	0	0	1	1	0	1	0
0	0	0	1	0	1	0	0	1	1
0	0	1	1	1	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0
			x=0			x=1			

Voor het afleiden van de logische vergelijkingen voor Y_1, Y_2 en Y_3 moet in de tabel worden nagegaan voor welke combinatie van y_1, y_2, y_3, x respectievelijk Y_1, Y_2 en Y_3 gelijk is aan 1.

Zo geldt voor Y_1:

$$Y_1 = \bar{y}_1\bar{y}_2\bar{y}_3\bar{x} + \bar{y}_1\bar{y}_2y_3\bar{x} + \bar{y}_1y_2\bar{y}_3\bar{x} + (\bar{y}_1y_2y_3\bar{x} + y_1y_2y_3\bar{x}) +$$
$$y_1y_2\bar{y}_3x + y_1\bar{y}_2y_3x + y_1\bar{y}_2\bar{y}_3x + (\bar{y}_1y_2y_3x + y_1y_2y_3x)$$

Hierbij kunnen de termen tussen haakjes als don't cares worden beschouwd zodat ze alleen worden meegenomen als hierdoor de logische functie eenvoudiger wordt.

Voor Y_1 wordt gevonden: $Y_1 = y_1x + \bar{y}_1\bar{x}$.

Met toewijzing B en toewijzing A worden voor Y_1, Y_2 en Y_3 de volgende logische vergelijkingen gevonden:

toewijzing B;

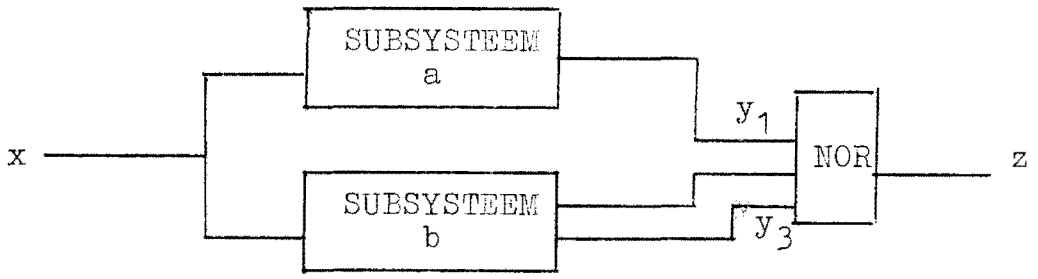
$$Y_1 = y_1x + \bar{y}_1\bar{x}$$
$$Y_2 = y_3\bar{x}$$
$$Y_3 = \bar{y}_2\bar{y}_3$$

toewijzing A;

$$Y_1 = \bar{y}_1\bar{y}_2y_3\bar{x} + y_2\bar{y}_3\bar{x} + y_2y_3x$$
$$Y_2 = \bar{y}_2x + \bar{y}_1\bar{y}_2\bar{y}_3 + y_1y_3$$
$$Y_3 = y_1x + y_2\bar{y}_3x + y_2y_3\bar{x} + \bar{y}_1\bar{y}_2\bar{x}$$

Voor toewijzing B zijn de logische vergelijkingen veel eenvoudiger omdat de functionele afhankelijkheid tussen de

toestand variabelen is gereduceerd.
Voor toewijzing B blijkt y_1 niet afhankelijk te zijn van y_2 en y_3. Ook blijkt dat y_2 en y_3 niet afhankelijk zijn van y_1. Het systeem kan worden verdeeld in twee onafhankelijke component systemen a en b die parallel geschakeld zijn. (zie figuur 5.1)



Figuur 5.1: Realisatie van machine 1 volgens toewijzing B.

Voor subsysteem a geldt in feite de volgende toestand tabel:

	input x	
	0	1
0,1,2	3,4,5	0,1,2
3,4,5	0,1,2	3,4,5

of y_1

	input x	
	0	1
1	0	1
0	1	0

Y_1

Voor subsysteem b geldt de volgende toestand tabel

	input x	
	0	1
0,5	2,3	2,3
1,4	0,5	2,3
2,3	1,4	1,4

of y_2y_3

	input x	
	0	1
10	00	00
01	10	00
00	01	01

Y_2Y_3

Voor variabele y_1 kan er geen onderscheid gemaakt worden tussen de toestanden 0,1 en 2 of 3,4 en 5.
Voor y_2y_3 bestaat er geen onderscheid tussen de toestanden 1,4 of 0,5 of 2,3.

Theorema 5.3 legt het verband tussen een y-toewijzing en een extended partition.
Volgens theorema 5.3 kan voor ieder element y_i een EP P_i

worden geconstrueerd. Voor toewijzing E geldt dan:

y_1: $P_1 = \{0,1,2;3,4,5\}$
y_2: $P_2 = \{0,5;1,2,3,4\}$
y_3: $P_3 = \{1,4;0,2,3,5\}$

Volgens definitie 5.4 is y_i een significant element want de EP P_i bevat 2 blokken. Het product $P_1.P_2.P_3 = \{0;1;2;3;4;5\}$ Voor subsysteem b moeten y_1 én y_2 worden gerealiseerd. Daarom moet er volgens theorema 5.1 een EP worden gevonden welke $\leqslant P_2$ en $\leqslant P_3$ is. Volgens theorema 5.2 voldoet hieraan het product van P_2 en P_3.

y_2y_3: $P_{23} = \{0,5;1,4;2,3\}$

Uit dit voorbeeld blijkt dat het splitsen van een machine in component machines kan leiden tot " economische"toestand toewijzingen .Intuitief kan worden vastgesteld dat de reductie van het aantal toestand variabelen en inputs waar de toestand variabelen van afhangen, de logische circuits vereenvoudigt. De grondidee bij de mathematische synthese is om methodes te vinden volgens welke toewijzingen kunnen worden geselecteerd, waarbij iedere variabele welke de nieuwe toestand beschrijft van zo weinig mogelijk variabelen die de oude toestand beschreven,afhangt. Een belangrijk mathematisch gereedschap is hierbij het concept van een EP-triplet. Er kunnen voorwaarden worden gevonden voor het bestaan van toestand toewijzingen met gereduceerde afhankelijkheid in termen van EP-triplets.

Definitie 5.5

Gegeven een systeem:

$$M=(S,T,I,O,d,1)$$

en twee EP's P en Q op S en een EP E op I. Dan is de geordende rij (E,P,Q) een EP-triplet, afgekort tot EPT, op M als en slechts als:

$$s \equiv t(P) \text{ en } x \equiv y \ (E)$$

impliceren:

$$d(s,x) \equiv d(s,y) \equiv d(t,x) \equiv d(t,y) \ (Q)$$

voor alle s,t in S en x,y in I waarbij d gedefinieerd is.

Dus het triplet (E,P,Q) is een EPT als de combinaties van blokken van E en P door M afgebeeld worden in Q, ofwel voor alle B(E) in E en B(P) in P is er een B(Q) in Q zodanig dat geldt:

$$d[B(P),B(E)] \leq B(Q)$$

Voorbeeld 7

	a	b	c	d
1	1	2	-	-
2	-	2	3	-
3	-	-	3	4
4	1	-	-	4

Stel $P=\{1,2,3;4\}$
$E=\{a,d;b,c\}$

Uit de toestand tabel blijkt dat het systeem voor de toestanden 1,2,3 bij inputs a,d overgaat in blok B(1) met toestanden 1,4 en bij inputs b,c overgaat in blok B(2) met toestanden 2,3. Het systeem gaat voor toestand 4 bij inputs a,d over in blok B(3) met toestanden 1,4 terwijl er voor de inputs b,c geen volgende toestand gedefinieerd is.
Met de gevonden blokken is een EP Q' te construeren. Dit houdt in dat voor de blokken van Q' moet gelden volgens definitie 5.1:

1. $B(i) \leq B(j)$ impliceert i=j
2. $\pm B(i) = S$

De blokken waarmee Q' moet worden bepaald zijn 1,4 ; 2,3 ; 1,4. De eerste eis heeft dus tot gevolg dat het derde blok kan worden weggelaten. De eis kan worden gezien als een Max-operatie op de blokken B(1)/B(3).
Aan de tweede eis wordt steeds voldaan omdat iedere P alle toestanden bevat en iedere E alle inputs.
Dus geldt:

$$Q' = \{B(1); B(2)\} = \{1,4; 2,3\}$$

Volgens definitie 5.5 zal iedere E,P en Q een EPT zijn als geldt:

$$Q \geq Q'$$

Zo zijn:

$(\{a,d;b,c\},\{1,2,3;4\},\{1,4;2,3\})$
$(\{a,d;b,c\},\{1,2,3;4\},\{1,2,4;2,3\})$
$(\{a,d;b,c\},\{1,2,3;4\},\{1,2,3,4\})$

allen EPT's op M.

Als bekend is in welk blok van P de toestand van de machine voorkomt dan kan voor ieder input blok van E worden bepaald in welk blok van Q de volgende toestand voorkomt. M.a.w. is er een bepaalde onbekendheid over de toestand waarin het systeem zich bevindt maar weten we alleen in welk blok van P de toestand voorkomt dan wordt bij een bepaalde E, de onbekendheid over de volgende toestand van het systeem gegeven door de EP Q . Van Q kan immers alleen de blok worden berekend welke de volgende toestand van P bevat.
Uit de toestand tabel blijkt dat P de rijen aangeeft welke 'gelijk' gemaakt moeten worden bij een bepaalde E en dat de EP Q de equivalente klassen van toestanden aangeeft welke voldoende zijn om de gelijkstelling van de rijen mogelijk te maken.

Zijn de triplets (E,P,Q) en (F,R,S) voor systeem M bekend dan kunnen hieruit nieuwe EPT's worden gegenereerd volgens het volgende theorema:

Theorema 5.4

Als (E,P,Q) en (F,R,S) EPT's op een systeem M zijn dan zijn ook:

(E.F,P.R,Q.S),
(E,P+R,Q+S) mits $E \leq F$,
(E+F,P,Q+S) mits $P \leq R$

EPT's op M.

Als het triplet (E,P,Q) een EPT is dan zullen de triplets die worden verkregen door E en/of P te verkleinen en/of Q te vergroten eveneens EPT's zijn. Dit kan formeel worden beschreven door het volgende theorema:

Theorema 5.5

Stel (E,P,Q) is een EPT op een systeem M. Stel $F \leq E$, $R \leq P$ en $Q \leq S$. Dan zijn de triplets:

(F,P,Q) (E,R,Q) (F,R,Q) (E,P,S) (F,P,S) (E,R,S) en (F,R,S)

eveneens EPT's op M.

Het bewijs van theorema 5.4 en 5.5 geschiedt aan de hand van de relatie $d[B(P),B(E)] \leq B(Q)$, die volgens definitie 5.5 voor een EPT geldt.

Voorbeeld 8

Volgens theorema 5.5 is voor een EPT (E,P,Q) op M en een EP $R \leq P$, (E,R,Q) ook een EPT op M.
Als $R \leq P$ dan is er voor iedere B(R) een B(P) zodat B(R) een deelverzameling is van B(P).
Als (E,P,Q) een EPT op M is geldt:

$$d[B(P),B(E)] \leq B(Q)$$

voor alle toestanden en inputs waarbij d gedefinieerd is.
Maar dan is ook:

$$d[B(R),B(E)] \leq B(Q)$$

zodat het triplet (E,R,Q) een EPT op M is.

Uit definitie 5.5, theorema 5.4 en theorema 5.5 blijkt dat er zeer vele EPT's voor een systeem M kunnen worden gevonden. Er zal daarom een speciale klasse van triplets worden gedefinieerd, de zogenaamde MMm triplets, waaruit alle andere triplets kunnen worden afgeleid. De afleiding kan geschieden door de 'secundaire' EP (Q) van het MMm triplet te vergroten en de 'primaire' EP's (I en P) van het MMm triplet te verkleinen.
De EPT's (E,P,Q) geven aan welke blokken van Q tegelijk uit blokken van E en P bepaald kunnen worden. De 'meeste informatie' wordt uit de 'minste informatie' verkregen als de blokken van E en P zo groot mogelijk zijn en de blokken van Q zo klein mogelijk zijn. Bestaat de EP Q uit blokken met slechts 1 toestand, dan is de volgende toestand van het systeem M eenduidig bepaald.
Bij een gegeven E en P kan nu een minimale Q bepaald worden zodanig dat (E,P,Q) een EPT is. Evenzo kan bij een gegeven P en Q,resp. E en Q, een maximale E, resp. P, bepaald worden zodat het triplet (E,P,Q) een EPT is. Voor alle EP's P,Q op S en E op I geldt dat:

(E,P,I) en (0,0,Q)

EPT's op M zijn. Hierbij is I de EP die slechts 1 blok bevat

(De eenheidspartitie) en O de EP waarbij ieder blok 1 element bevat (de nulpartitie). Maar dan is er volgens theorema 5.4 een minimale Q zodanig dat (E,P,Q) een EPT is en een maximale P, resp. E. zodat (E,P,Q) een EPT is.

Definitie 5.6

Als P en Q EP's op S zijn en E is een EP op I dan is:

$$m(E,P) = \prod \{Q(i) \mid (E,P,Q(i)) \text{ is een EPT op } M\}$$
$$M(E,Q) = \sum \{P(i) \mid (E,P(i),Q) \text{ is een EPT op } M\}$$
$$M(P,Q) = \sum_{i} \{E(i) \mid (E(i),P,Q) \text{ is een EPT op } M\}$$

m() kan als operator opgevat worden die de kleinst mogelijke EP Q geeft bij gegeven E en P (m voor minimum). Evenzo kan M() als operator opgevat worden die bij gegeven E en Q, resp. P en Q, de maximale P, resp. E, geeft (M voor maximum).

Bij een gegeven E en P beschrijft m(E,P) de maximale informatie welke we over de volgende toestand kunnen berekenen als alleen P bekend is (d.w.z. het blok wat de aanwezige toestand van systeem M bevat). Voor een gegeven Q beschrijft M(E,Q) de informatie die minimaal bekend moet zijn over de huidige toestand om Q voor de volgende toestand te kunnen bepalen.

Definitie 5.7

Het EPT (E,P,Q) is een MMm triplet dan en slechts dan als:

1. $P = M(E,Q)$
2. $E = M(P,Q)$
3. $Q = m(E,P)$

Het volgende theorema beschrijft een methode om M-EP's en m-EP's te bepalen als twee EP's gegeven zijn.

Theorema 5.6

Gegeven een systeem M = (S,T,I,O,d,1), een EP E op I en de EP's P en Q op S. Dan is:

$$M(E,Q) = \sum \{P \mid m(E,P) \leq Q\}$$

en:

$$M(P,Q) = \sum \{E \mid m(E,P) \leq Q\}$$

en:

$$m(E,P) = \sum \{m(F,P) \mid F \leq E\} = \sum \{m(E,R) \; R \leq P\}$$

Voor het bewijs van theorema 5.6 (en alle andere theorema's die in dit hoofdstuk worden genoemd) zij verwezen naar literatuuropgave 1 hoofdstuk 4.

Om bij een tweetal EP's eenvoudig de bijbehorende M-EP, resp. m-EP, te bepalen worden eerst alle mogelijke basis-EP's met de daarbij behorende m-EP's bepaald. Een basis-EP is in dit verband een EP waarbij 1 blok i elementen ($1 \leq i \leq n$ bij verzamelingen met n elementen) bevat en alle andere blokken 1 element bevatten. De verzameling van alle basis-EP's omvat nu alle blokken die voor kunnen komen, zodat door sommatie van een aantal basis-EP's iedere EP geconstrueerd kan worden. Door sommatie van die EP's die aan de gestelde voorwaarden voldoen wordt dan de M-EP, resp. m-EP, bepaald.

Voorbeeld 9

Beschouw het systeem in onderstaande sequentietabel:

	a	b	c
1	1	2	-
2	-	2	3
3	1	-	3

Als $M(E,Q) = M[\{a,c;b\}, \{1,3;2,3\}]$ bepaald moet worden zijn de benodigde basis-EP's:

$$m[\{a,c;b\}, \{1,2,3\}] = \{1,3;2\}; \quad m(E_1,P_1) = Q_1$$
$$m[\{a,c;b\}, \{1,2;3\}] = \{1,2;3\}; \quad m(E_2,P_2) = Q_2$$
$$m[\{a,c;b\}, \{1,3;2\}] = \{1,3;2\}; \quad m(E_3,P_3) = Q_3$$
$$m[\{a,c;b\}, \{1;2,3\}] = \{1;2;3\}; \quad m(E_4,P_4) = Q_4$$
$$m[\{a,c;b\}, \{1;2;3\}] = \{1;2;3\}; \quad m(E_5,P_5) = Q_5$$

$$M(E,Q) = \sum_{i=1}^{5} \{P_i \mid m(E,P_i) \leq Q\} = \sum_{i=1}^{5} \{P_i \mid Q_i \leq Q\}$$

Dus: $M[\{a,c;b\}, \{1,3;2,3\}] = \{1,2,3\}$

Voor de bepaling van m-EP's kan volgens theorema 5.6 ook worden uitgegaan van de basis-EP's. De gewenste m-EP's kunnen echter veel eenvoudiger direct uit de sequentietabel worden afgeleid, zonder gebruik te maken van de basis-EP's. Het aantal basis-EP's neemt namelijk zeer sterk toe met het aantal elementen: voor n elementen(toestanden of inputs) zijn er 2^n-n basiselementen.

Voorbeeld 10

In voorbeeld 7 zijn een aantal triplets afgeleid voor het systeem M met E=$\{a,d;b,c\}$ en P=$\{1,2,3;4\}$. Hierbij bleek dat de EP Q' de kleinste EP was zodanig dat voor Q = Q', E,P,Q nog een triplet vormden. Dus geldt m(E,P)=Q'.

Voorbeeld 11

Beschouw het systeem M beschreven in onderstaande sequentietabel:

	0	1
1	2	5
2	3	4
3	1	4
4	3	2
5	4	-

Stel E=$\{0,1\}$

P=$\{1,2,3;2,3,4;4,5\} = \{B_1;B_2;B_3\}$

Nu geldt $d(B_1,0)$= 1,2,3
$d(B_1,1)$= 4,5
$d(B_2,0)$= 1,3
$d(B_2,1)$= 2,4
$d(B_3,0)$= 3,4
$d(B_3,1)$= 2

m(E,P) is dan gelijk aan:
m(E,P)=Max (1,2,3;4,5;1,3;2,4;3,4;2) =$\{1,2,3;4,5;2,4;3,4\}$

De MMm triplets vormen een compacte manier om de structurele informatie over een sequentiele machine te representeren. Voor het genereren van alle MMm triplets voor het CS kunnen de inputverzamelingen E worden opgesplitst volgens de input-variabelen welke gegeven zijn. Voor drie input-variabelen a,b en c ontstaan dan de volgende verzamelingen:

a : 0XX ; 1XX
b : X0X ; X1X
c : XX0 ; XX1
ab : 00X ; 01X ; 10X ; 11X
ac : 0X0 ; 0X1 ; 1X0 ; 1X1
bc : X00 ; X01 ; X10 ; X11
abc : 000 ; 001 ; 010 ; 100 ; 101 ; 110 ; 111

Voor m input variabelen ontstaan 2^m-1 inputverzamelingen. Bij iedere input verzameling zullen voor de toestanden alle basis-EP's moeten worden bepaald. Voor n toestanden zijn er (2^n-n)basis-EP's. Uit de sequentie tabel volgt dan m(E,P). Het aantal triplets waaruit de MMm triplets moeten worden gevormd is dus gelijk aan $(2^m-1)(2^n-n)$. De M(E,Q) volgt verder uit de eerste formule van theorema 5.6. Dit vereist ongeveer $\frac{1}{2}(2^m-1)^2(2^n-n)^2$ sommaties.
De bepaling van MMm triplets zal kunnen geschieden m.b.v. een computerprogramma. Bovenstaande berekening toont echter aan dat deze bepaling beperkt zal moeten blijven tot systemen met slechts enkele toestanden en inputs.

Er zal nu worden ingegaan op de toepassing van triplets bij het toestands-toekenning probleem. Hierbij zullen noodzakelijke en voldoende voorwaarden worden beschreven voor 'gereduceerde' afhankelijkheid.

Theorema 5.7

Gegeven een systeem M=(S,T,I,O,d,l) en de EP's P,Q op S en E op I. Dan is:

$$m(E,P) \leq Q \;;\; P \leq M(E,Q) \;;\; E \leq M(P,Q)$$

als en slechts als er een eenduidige partiele functie f:

$$f: P * E \dashrightarrow Q$$

bestaat zodat:

$$f\left[B(P,s),B(E,x)\right] = B\left[Q,d(s,x)\right]$$

voor alle s in S en x in I waarbij d(s,x) gedefinieerd is.

B(P,s) geeft een blok van P aan dat de toestand s bevat. B(E,x) geeft een blok van E aan dat de input x bevat. Informeel zegt het theorema, dat als de toestand EP P genoeg informatie bevat om uit de input de volgende blok van Q te berekenen en de input EP E bevat genoeg informatie om uit de aanwezige toestand de volgende blok van Q te berekenen, dan bevatten P en E samen genoeg informatie om de volgende blok van Q te berekenen. De functie f duidt een blok van Q aan en de conditie $f\left[B(P,s),B(E,x)\right]=B\left[Q,d(s,x)\right]$ geeft aan dat f

het blok van Q noemt dat de volgende toestand bevat. De EP's E,P en Q vormen een triplet.
Het effect van theorema 5.7 is de vaststelling dat de input afhankelijkheid en de toestand afhankelijkheid tegelijk kunnen worden gereduceerd.
Voor het bewijs van theorema 5.7 zie lituratuuropgave 1 hoofdstuk 4.

Theorema 5.7 is uit te breiden tot een verzameling EP's P(i) op S , een aantal EP's E(j) op I en de EP Q op S.

Theorema 5.8

Gegeven een systeem M= S,T,I,O,d,l , een aantal EP's P(i) op S, een aantal EP's E(j) op I en de EP Q op S. Dan is:

$$m\left[\bigsqcap_j E(j), \bigsqcap_i P(i)\right] \leq Q$$

en:

$$\bigsqcap_i P(i) \leq M\left[\bigsqcap_j E(j), Q\right] \text{ en } \bigsqcap_j E(j) \leq M\left[\bigsqcap_i P(i), Q\right]$$

als en slechts als er een partiele functie f:

$$f: \left[\overset{*}{\underset{i}{}} P(i) , \overset{*}{\underset{j}{}} E(j)\right] \rightarrow Q$$

bestaat zodat:

$$f\left[\overset{*}{\underset{i}{}} B(P(i),s), \overset{*}{\underset{j}{}} B(E(j),x)\right] = B\left[Q, d(s,x)\right]$$

voor alle s in S en x in I waarbij d(s,x) gedefinieerd is.

Als we zeggen dat een variabele slechts van een zeker aantal variabelen afhangt,dan betekent dit dat de variabele kan worden berekend uit deze set en de input, onafhankelijk van de waarde van de andere toestand variabelen. Het volgende theorema toont aan dat het begrip van afhankelijkheid van variabelen equivalent is met het bestaan van bepaalde algebraische relaties tussen de partities, welke geassocieerd zijn met de variabelen.

Theorema 5.9

Stel dat voor een systeem M = (S,T,I,O,d,l), toestandvariabelen $\{ y(i)\}$ zijn toegewezen volgens EP's $\{ P(i)\}$, inputvariabelen $\{ x(j)\}$ zijn toegewezen volgens EP's $\{E(j) \}$,

en dat een G en een H gegeven zijn zodanig dat

$$G \leq \{y(i)\} \text{ en } H \leq \{x(j)\}$$

Dan kan $\ddot{y}(k)$ als functie van de variabelen in $G \pm H$ bepaald worden als geldt:

$$m\left[\overline{\lceil_G} P(i), \overline{\lceil_H} E(j)\right] \leq P(k)$$

of:

$$\overline{\lceil_G} P(i) \leq M\left[\overline{\lceil_H} E(j), P(k)\right] \text{ en } \overline{\lceil_H} E(j) \leq M\left[\overline{\lceil_G} P(i), P(k)\right]$$

Bewijs. Er is een functie Y)k):

$$Y(k): \left[\overset{*}{}_G y(i), \overset{*}{}_H x(j)\right] \rightarrow y(k)$$

nodig zodat:

$$Y(k)\left[\overset{*}{}_G y(i(s)), \overset{*}{}_H x(j(a))\right] = y\left[k(d(s,a))\right]$$

voor alle s in S en a in I waarbij d(s,a) gedefinieerd is. Het bereik van de variabelen correspondeert op eenduidige wijze metde EP's volgens welke zij zijn toegewezen, dus kan Y(k) beschouwd worden als de functie:

$$Y(k): \left[\overset{*}{}_G P(i), \overset{*}{}_H E(j)\right] \rightarrow P(k)$$

zodat:

$$Y(k)\left[\overset{*}{}_G B(P(i),s), \overset{*}{}_H B(E(j),a)\right] = B\left[P(k), d(s,a)\right]$$

voor alle s in S en a in I waarbij d(s,a) gedefinieerd is. Volgens theorema 5.8 bestaat deze functie als aan de gestelde voorwaarden wordt voldaan.

Uit economische overwegingen dienen de verzameling G en H zo klein mogelijk te zijn omdat dan de functies zo weinig mogelijk variabelen bevatten en dan de realisatie zo goedkoop mogelijk is.
De eis $m\left[\overline{\lceil_G} P(i), \overline{\lceil_H} E(j)\right] \leq P(k)$ houdt in dat het triplet $\left[\overline{\lceil_H} E(j), \overline{\lceil_G} P(i), P(k)\right]$ een EPT op het CS is.

Uit de MMm triplets kunnen een aantal EP's P(i) van twee blokken worden afgeleid zodanig dat de doorsnede van deze EP's nul is. Voor verminderde afhankelijkheid worden die P(i) gekozen die voldoen aan de relaties uit theorema 5.9. Als de input variabelen vastliggen zal gezocht worden naar een $M\lceil \lceil_H E(j),P(k)\rceil$ welke groter of gelijk is aan het product van slechts een kleine subset van de partities P(i) (subset G). Hoe groter $M\lceil \lceil_H E(j),P(k)\rceil$ des te beter zal hieraan kunnen worden voldaan.

6. ONTWERP VAN HET CONTROL SYSTEM

In hoofdstuk 5 is een formele methode gegeven om een systeem te ontwerpen door uit te gaan van de MMm triplets. De grote moeilijkheid hierbij is het vinden van een geschikte toewijzing van de toestand variabelen. Door het grote aantal triplets zullen er namelijk vele toewijzingen met verminderde afhankelijkheid mogelijk zijn welke in feite allen moeten worden nagelopen. In dit hoofdstuk zal het CS worden ontworpen door een bepaalde keuze te doen voor de toewijzing van de toestand variabelen. De keuze geeft geen garantie voor het vinden van het 'meest geschikte ontwerp' maar wel een garantie dat de outputs gerealiseerd kunnen worden.

Volgens definitie 4.1.1 zijn de outputs van het CS alleen een functie van de toestanden van het CS. Voor deze functies kan nu een een-op-een afbeelding worden gekozen. Dit betekent dat iedere output variabele overeenkomt met een toestand variabele. Dit heeft weer tot gevolg dat de eisen voor de toestand variabelen overeenkomen met de eisen van de output variabelen. Door deze keuze liggen de toestandvariabelen vast en zal het circuit wat normaal de outputfunctie afleidt uit de sequentiele schakeling, reduceren tot doorverbindingen.

Voorbeeld 1

Beschouw het systeem beschreven in onderstaande tabel

$d(s,x)$	s	A	B	p(1)	p(2)
1	1	0	X	0	1
1	3	1	X	0	1
2	1	0	X	0	0
2	2	0	X	0	0
3	2	0	1	1	0
3	3	0	1	1	0

Als iedere outputvariabele overeenkomt met een toestandvariabele, dan zal de toewijzing van de toestandvariabelen voor dit systeem als volgt zijn:

toestand 1: 01
toestand 2: 00
toestand 3: 10

Het Control System zal ontworpen zijn als de outputs gerealiseerd worden. Hiertoe moet eerst worden nagegaan of bij deze keuze van toestandvariabelen de outputs gerealiseerd kunnen worden. Dit kan worden nagegaan aan de hand van relaties welke in hoofdstuk 5 ontwikkeld zijn. Als de outputs niet kunnen worden gerealiseerd zullen er secundaire toestand variabelen(c.q. nieuwe outputs) moeten worden ingevoerd net zo lang tot de realisatie van alle outputs mogelijk is.

Er zal nu worden nagegaan onder welke condities de outputs worden gerealiseerd en hoe eventueel de secundaire toestandvariabelen kunnen worden bepaald.

Definitie 6.1

De EP van een executie voorwaarde P(k) op de verzameling toestanden S van het CS is de verzameling P(k):

$$P(k) = \text{Max}\{ B^1(i), B^0(i) \}$$

met:

$$B^1(i) = \{ s \text{ in } S \mid \text{toestand van } p(i) \text{ is 1 of X} \}$$
$$B^0(i) = \{ s \text{ in } S \mid \text{toestand van } p(i) \text{ is 0 of X} \}$$

Definitie 6.2

Stel p(k) met $1 \leq k \leq r$ is de verzameling van alle EP's van de executie voorwaarden, dan zijn de elementen van P(0):

$$P(0) = \underset{k=1}{\overset{k=r}{\top}} P(k)$$

de deelverzamelingen toestanden van het CS die onderscheiden moeten kunnen worden om alle P(k) te realiseren.

Omdat $P(0) \leq P(k)$ met $1 \leq k \leq r$ geldt immers volgens theorema 5.1 dat P(0) alle P(k) realiseert. Dus geven de blokken van P(0) aan welke toestanden van het CS van elkaar onderscheiden moeten worden.

De eisen voor output k van het CS worden gegeven door de EP P(k) (definitie 6.1). De EP dient nu te worden gerealiseerd door de EP R(k) op S voor toestandvariabele r(k).(Zie theorema 5.1) R(k) moet dus voldoen aan:

$$R(k) \leq P(k)$$

Om de toestandvariabele r(k) als functie van een aantal toestandvariabelen en inputvariabelen te kunnen bepalen moet volgens theorema 5.9 het triplet

$$[\overline{\top_H\top}\, E(j), \overline{\top_G\top}\, R(i), R(k)]$$

een EPT op het CS zijn.
Omdat $R(i) \leq P(i)$, is zeker aan de voorwaarde voldaan als het triplet

$$[\overline{\top_H\top}\, E(j), \overline{\top_G\top}\, P(i), R(k)]$$

een EPT op het CS is.
Omdat P(k) een bovengrens voor R(k) is wordt R(k) voldoende nauwkeurig bepaald als het triplet:

$$[\overline{\top_H\top}\, E(j), \overline{\top_G\top}\, P(i), P(k)]$$

een EPT op het CS is. Voor R(k) wordt dan gevonden:

$$R(k) = m[\overline{\top_H\top}\, E(j), \overline{\top_G\top}\, P(i)]$$

Theorema 6.1

Gegeven een systeem M = (S,T,I,O,d,l) en stel:

$$T(H,G) = m[\overline{\top_H\top}\, E(j), \overline{\top_G\top}\, P(i)]$$

met $G \leqq \{P(i)\}$ en $H \leqq \{E(j)\}$. Dan bestaat er voor toestandvariabele k een functie van de variabelen in $G \pm H$ als:

$$T(H,G) \leq P(k)$$

Bewijs. Het triplet:

$$[\overline{\top_H\top}\, E(j), \overline{\top_G\top}\, P(i), T(H,G)]$$

is een EPT op M. Als $T(H,G) \leq P(k)$ is dus ook het triplet:

$$[\overline{\top_H\top}\, E(j), \overline{\top_G\top}\, P(i), P(k)]$$

een EPT op M en dus bestaat er volgens theorema 5.9 voor toestandsvariabele k een functie van de variabelen in $G \pm H$.

Volgens theorema 6.1 is T(H,G) gedefinieerd als:

$x \equiv y [\overline{\top_H\top}\, E(j)]$ en $s \equiv t [\overline{\top_G\top}\, P(i)]$ impliceert

$$d(s,x) \equiv d(t,y) [T(H,G)]$$

voor alle s,t in S en x,y in I waarbij d gedefinieerd is.

Er zal volgens definitie 6.2 voor alle P(k) een functie bestaan als geldt:

$$T(H,G) \leq P(0)$$

De deelverzamelingen G en H waarvoor de kleinste T(H,G) wordt gevonden zijn respectievelijk P(0) en E(0) waarbij E(0) de 0-EP op I is. Dit geeft de volgende eis voor het bestaan van een outputfunctie voor iedere output:

$$m\left[E(0),P(0)\right] \leq P(0)$$

Voorbeeld 2

Beschouw het systeem in onderstaande tabel:

d(s,x)	s	A	B	p(k)
1	1	X	0	0
1	3	X	0	0
2	1	X	1	1
2	2	0	1	1
3	2	1	1	1
3	3	1	1	1

In dit systeem is de EP voor de executie voorwaarde:

$$P(k) = \{1;2,3\} = P(0)$$

Om na te gaan of de output kan worden afgeleid uit de toestandvariabelen en inputvariabelen, zal moeten worden gecontroleerd of $m\left[E(0),P(0)\right] \leq P(0)$.

Er kan dus worden nagegaan of $d(x,B(P(0))) \leq B(P(0))$ of $\text{Max}\{d(x,B(P(0)))\} \leq P(0)$.

Dit kan systematisch geschieden volgens onderstaande tabel:

x	B(P(0))	d(B(P(0)),x)
00	2,3	1
01	2,3	2
10	2,3	1
11	2,3	3
00	1	1
01	1	2
10	1	1
11	1	2

Uit de tabel volgt $T(H,G) = \text{Max}\,(1;2;1;3;1;2;1;2) = \{1;2;3\}$ zodat $T(H,G) \leq P(0)$.

Het is mogelijk dat met de gegeven toestandvariabelen (output-variabelen) niet aan de gestelde voorwaarden wordt voldaan, d.w.z. er wordt een T(H,G) gevonden die niet kleiner of gelijk is dan P(O). Om nu T(H,G) te verkleinen worden extra toestand variabelen ingevoerd. Er kan een EP Q worden bepaald welke de eisen geeft waaraan de secundaire variabele moet voldoen.
De invoering van een extra toestandvariabele geeft ook een extra outputvariabele. De eis voor het bestaan van alle output functies is nu:

$$m\left[E(0),\ P'(0)\right] \leq P'(0) \quad \text{met } P'(0)=Q.P(0)$$

Hierbij zal Q bepaald moeten worden. Het is zinvol om voor Q de grootste EP te gebruiken omdat deze het eenvoudigste te realiseren is.
Om Q te kunnen bepalen door een computerprogramma zal een algorithme worden gedefinieerd, dat alle secundaire toestand variabelen genereert. In het algorithme wordt eerst een secundaire toestand variabele Q berekend zodanig dat geldt:

$$m\left[E(0),Q.P(0)\right] \leq P(0)$$

Daarna wordt gecontroleerd of dan ook

$$m\left[E(0),\ P'(0)\right] \leq P'(0) \text{ met } P'(0)=Q.P(0)$$

Wordt aan deze voorwaarde niet voldaan dan zal een volgende secundaire toestandvariabele worden gegenereerd zodanig dat geldt

$$m\left[E(0),\ Q'.P'(0)\right] \leq P'(0)$$

Hierbij is Q' de volgende secundaire toestandvariabele.
Daarna wordt weer gecontroleerd of

$$m\left[E(0),\ Q'.P'(0)\right] \leq Q'.P'(0) \text{ enzovoorts.}$$

Het proces van de bepaling van secundaire toestand variabelen is eindig om dat uiteindelijk de O-EP op S wordt bereikt.

Theorema 6.2

Gegeven een systeem M=(S,T,I,O,d,1), een EP $P = \{B(P,i)\}$ op S.
Dan is de EP Q:

$$Q = \text{Max}\left\{ \underset{x,j}{\cdot\cdot} \ \underset{i}{\text{Max}} \left\{ B(x,j,i),\ s \mid s \text{ in } S \right\} \right\}$$

met:

$$B(x,j,i) = \left\{ S - \{s\} \mid d(s,x) \text{ niet in } B(P,i) \text{ als } s \text{ in } B(P,j) \right\}$$

de grootste EP op S die voldoet aan $m\left[O,\ Q.P\right] \leq P$

Bewijs

Door het toevoegen van alle blokken met ieder 1 toestand en door de Maxoperatie is Q een EP op S.
Uit de constructie van B(x,j,i) volgt dat B(x,j,i) een deelverzameling van S is die voldoet aan:

$$d\left[\ B(x,j,i) \underline{.}\ B(P,j),x\right] \leqq B(P,i) \text{ voor een } x,i,j$$

Maar dan bevat $\mathrm{Max}\{\ B(x,j,i)\}$, voor een x,j en verenigd voor alle i, de deelverzamelingen van S die voldoen aan:

$$d\left[\ B\left[\ \mathrm{Max}\{B(x,j,i)\}\right]\ \underline{.}\ B(P,j),x\right] \leqq B(P,i) \text{ voor een } x,j$$

en bevat $\mathrm{Max}\{\ \underset{i}{\underline{.}}\ \mathrm{Max}\{\ B(x,j,i)\}\}$, verenigd voor alle i en doorsneden voor alle x,j, de deelverzamelingen van S die voldoen aan:

$$d\left[\ B\left[\ \mathrm{Max}\{\ \underset{x,j}{\underline{.}}\ \underset{i}{\mathrm{Max}}\{B(x,j,i)\}\}\right]\ \underline{.}\ B(P,j),x\right] \leqq B(P,i)$$

Dus is het triplet:

$$\left[E(0),\ \mathrm{Max}\{\ \underset{x,j}{\underline{.}}\ \{\underset{i}{\mathrm{Max}}\ B(x,j,i)\}\ ,s\}\ .P,P\right] = \left[0,Q.P,P\right]$$

een EPT op M en dus is:

$$m\left[0,Q.P\right] \leqq P$$

Als een blok van Q vergroot wordt met een t in S bevat Q het blok:

$$\{\ \mathrm{Max}\{\ \underset{x,j}{\underline{.}}\ \underset{i}{\mathrm{Max}}\{B(x,j,i)\}\}\ ,t\} \text{ voor een } x,j,i$$

Dan is:

$$d\left[\{\mathrm{Max}\{\ \underset{x,j}{\underline{.}}\ \underset{i}{\mathrm{Max}}\{\ B(x,j,i)\}\}\ ,t\}\ \underline{.}\ B(P,j),x\right] =$$
$$= d\left[\ \mathrm{Max}\{\ \underset{x,j}{\underline{.}}\ \underset{i}{\mathrm{Max}}\{B(x,j,i)\}\}\ \underline{.}\ B(P,j),x\right] \pm d\left[\{t\}\ \underline{.}\ B(P,j),x\right]$$
$$= B(P,i) \pm d\left[\{t\}\ \underline{.}\ B(P,j),x\right]$$

Nu moet $d\left[\{t\}\ \underline{.}\ B(P,j),x\right]$, voor alle B(P,j), een element van B(P,i) zijn om te voldoen aan de eis:

$$m\left[\ 0,Q.P\right] \leqq P$$

Hieraan wordt voldaan als:

$\{t\}\ \underline{.}\ B(P,j)$ is leeg voor alle B(P,j)

of:

$$d\left[\{t \mid t \text{ in } B(P,j)\}\ ,x\right] \text{ in } B(P,i) \text{ voor alle } B(P,j)$$

Aan de eerste voorwaarde wordt alleen voldaan als t geen element van S is. Aan de tweede voorwaarde wordt alleen voldaan als B(P,i)=S, maar dan is Q de I-EP op S, zodat Q niet vergroot kan worden. Hieruit volgt dat Q de grootst mogelijke EP is.

Voorbeeld 3

Beschouw het systeem beschreven in onderstaande tabel:

d(s,x)	s	A	B	p(k)
1	1	0	X	0
1	3	1	X	0
2	1	0	X	1
2	2	0	X	1
3	2	0	1	0
3	3	0	1	0

In dit systeem is de EP voor de executie-voorwaarde:

$$P(k) = \{1,3;2\}$$

Voor T(H,G) voor alle input- en toestandvariabelen wordt gevonden:

$$T(H,G) = \{1,3;2,3\}$$

Er zijn dus niet voldoende toestandvariabelen om p(k) te kunnen realiseren. Als B(x,j,i) wordt bepaald, wordt gevonden:

B(P,j)	{1,3}		{2}	
B(P,i)	{1,3}	{2}	{1,3}	{2}
x= 00	{2,3}	{2,3}	{1,3}	{1,2,3}
x= 01	{2,3}	{2}	{1,2,3}	{1,3}
x= 10	{1,2,3}	{1,2}	{1,2,3}	{1,2,3}
x= 11	{1,2,3}	{1,2}	{1,2,3}	{1,2,3}

Uit de tabel volgt:

$$\max_i B(x,j,i) = \begin{matrix} \{2,3\} & \{1,2,3\} \\ \{2,3\} & \{1,2,3\} \\ \{1,2,3\} & \{1,2,3\} \\ \{1,2,3\} & \{1,2,3\} \end{matrix}$$

Dus geldt:

$$Q = \{1;2,3\}$$

Het product $Q.P(k) = \{1;2;3\}$

Omdat dit de nulpartitie op S is, zal er een functie bestaan voor zowel p(k) als q(1).

In het voorgaande voorbeeld bevat Q(i) 2 blokken, zodat Q(i) direct als EP voor de secundaire variabelen gebruikt kan worden. In het algemeen zal Q(i) meerdere blokken bevatten. Dan dienen een aantal 2-bloks EP's voor de secundaire variabelen geconstrueerd te worden die gezamelijk Q(i) realiseren.

Definitie 6.3

Stel de EP Q(i) op S geeft de eisen voor de secundaire variabelen. Dan is de verzameling EP's R(j) op S de verzameling EP's voor de secundaire variabelen als en slechts als:

1. $\prod_j R(j) \leq Q(i)$ voor alle j
2. Iedere R(j) bevat 2 blokken
3. $R(k) \leq R(l)$ impliceert l=k

De tweebloks EP's kunnen worden afgeleid van Q(i).
Hierbij moeten de blokken van Q(i) zodanig worden samengenomen dat twee blokken ontstaan die samen alle toestanden bevatten en waarbij de ene blok geen deelverzameling is van de andere blok.

Voorbeeld 4

Stel $Q(i)=\{1,2;1,3;2,4;3,4\}$

Hieruit zijn de volgende tweebloks EP's R(j) af te leiden:

$R(1)=\{1,2,3;2,3,4\}$
$R(2)=\{1,2,4;1,3,4\}$

R(1) en R(2) realiseren gezamelijk Q(i): Q(i)= R(1).R(2)

Bij het voorgaande proces is er voor gezorgd dat de outputs bepaald kunnen worden uit de toestand- en inputvariabelen, d.w.z. er is voor gezorgd dat:

$$T(H,G) \leq P(k) \text{ voor alle } k$$

Hierbij beschreven H en G respectievelijk alle input- en toestandvariabelen.
Voor iedere P(k) zal nu een T(H,G) moeten worden bepaald zodanig dat:

$$T(H,G) \leq P(k)$$

waarbij de verzamelingen G en H minimaal zijn. De outputs zijn dan van een minimaal aantal inputvariabelen en toestandvariabelen afhankelijk.

Voor zowel de inputs als de toestanden zijn de binaire variabelen gegeven. Door in de bij de simulatie gegenereerde tabel de momentane toestand te beschrijven met de toestanden van de toestandvariabelen gaat deze tabel over in een normale functietabel (vergelijk figuur 3.1.4/3.1.7 uit hoofdstuk 3). De outputfuncties kunnen nu met iedere methode voor het realiseren van combinatorische netwerken, die don't cares toestaan, gerealiseerd worden.

Voorbeeld 5

Beschouw het systeem beschreven in voorbeeld 2. Voor dit systeem bleek dat p(k) kon worden gerealiseerd.
Codering van de toestanden volgens de outputvariabele geeft voor toestand 1: 0
toestand 2,3: 1
Dit geeft de volgende functietabel:

	A	B	P	p
1	X	0	0	0
2	X	0	1	0
3	X	1	0	1
4	0	1	1	1
5	1	1	1	1
6	1	1	1	1

Uit de tabel volgt: $p = B.\bar{P} + \bar{A}.B.P + A.B.P + A.B.P = B$

Het afleiden van de outputfuncties kan ook geschieden door gebruik te maken van EP's. Hiervoor worden weer de toestanden met de outputvariabelen gecodeerd waarna de functietabel wordt bepaald. De regels van de functietabel worden genummerd. Met deze nummering kunnen EP's worden gecreëerd zoals is beschreven in theorema 5.3. Er zal nu worden gezocht naar een minimum aantal EP's E(j) die voldoen aan:

$$\prod_i E(j) \leq P(j)$$

waarbij het aantal i zo klein mogelijk moet zijn. Het aantal i geeft het aantal variabelen waar de output p(j) van afhangt.

Voorbeeld 6

In voorbeeld 5 kan voor inputvariabele B en output k als EP's: $E = \{1,2;3,4,5,6\}$ en $p = \{1,2;3,4,5,6\}$ worden afgeleid. Omdat $p \leq B$ kan de output p door 1 inputvariabele worden gere-

aliseerd. Er geldt p=B.

Algemeen zullen alle paren EP's E(j) en productparen bepaald moeten worden. Deze verzameling representeert alle Boolse functies die met een aantal elementen e(i) geconstrueerd kunnen worden. Hierna kan door de ontwerper uit deze verzameling een EP worden gekozen welke kleiner of gelijk is aan de output-EP en welke het resultaat is van het product van een minimaal aantal EP's.

Voorbeeld 6

Beschouw het systeem beschreven in onderstaande tabel:

d(s,x)	s	A B C	x y z
1	1	0 X X	0 0 0
1	5	0 0 X	0 0 0
2	1	1 0 X	0 0 0
2	2	1 0 X	0 0 0
3	2	1 1 0	X 1 0
3	3	1 1 0	X 1 0
4	2	1 1 1	X 0 1
4	4	1 1 1	X 0 1
5	3	1 0 X	1 X X
5	4	1 0 X	1 X X
5	5	1 0 X	1 X X

Hierbij worden de volgende EP's P(k) gevonden:

$P(x) = \{1,2,3,4;3,4,5\}$

$P(y) = \{1,2,4,5;3,5\}$

$P(z) = \{1,2,3,5;4,5\}$

$P(0) = \{1,2;3,5;4,5\}$

Als T(H,G), voor alle input- en toestandvariabelen, bepaald wordt dan wordt gevonden:

$T(H,G) = \{1;2;3;4;5\}$

zodat geen extra toestandvariabelen nodig zijn. Als de toestanden met de outputvariabelen gecodeerd worden ontstaat de volgende functie-tabel:

regel	A	B	C	X	Y	Z	x	y	z
1	0	X	X	0	0	0	0	0	0
2	0	0	X	1	X	X	0	0	0
3	1	0	X	0	0	0	0	0	0
4	1	0	X	0	0	0	0	0	0
5	1	1	0	0	0	0	X	1	0
6	1	1	0	X	1	0	X	1	0
7	1	1	1	0	0	0	X	0	1
8	1	1	1	X	0	1	X	0	1
9	1	0	X	X	1	0	1	X	X
10	1	0	X	X	0	1	1	X	X
11	1	0	X	1	X	X	1	X	X

Voor de EP's voor A,B,C,X,Y,Z op de verzameling regelnummers wordt gevonden:

A= {1,2;3,4,5,6,7,8,9,10,11}
B= {1,2,3,4,9,10,11;1,5,6,7,8}
C= {1,2,3,4,5,6,9,10,11;1,2,3,4,7,8,9,10,11}
X= {1,3,4,5,7,8,9,10,11;2,6,9,11,8}
Y= {1,2,3,4,5,7,8,10,11;2,6,9,11}
Z= {1,2,3,4,5,6,7,9,11;2,8,10,11}

De EP's voor de outputs x,y,z op de verzameling regelnummers zijn:

x= {1,2,3,4,5,6,7,8;5,6,7,8,9,10,11}
y= {1,2,3,4,7,8,9,10,11;5,6,9,10,11}
z= {1,2,3,4,5,6,9,10,11;7,8,9,10,11}

Deze EP's worden gerealiseerd door:

A.B.X.Y.Z ≤ x
A.B.C ≤ y
A.B.C ≤ z

Deze resultaten zijn in de functietabel gemakkelijk te verifiëren. Zo is bijvoorbeeld output y gelijk aan 1 voor regel 5 en regel 6. In deze regels zijn A en B beiden 1. Dit zijn ze ook in regel 7 en 8. Om regel 5,6 van regel 7,8 te onderscheiden kan input C worden gebruikt. (C=0 in regel 5,6 en C=1 in regel 7,8)

De bijbehorende numerieke functies zijn:

X= B (10001,10010,10100,10101,10110,10111) OF A,B,C,X,Y,Z
Y= B(110) OF A,B,C
Z= B(111) of A,B,C

Zie voor de notatie van de numerieke functies hoofdstuk 3.4.

Algemeen kan het CS worden opgevat als een serieschakeling van twee 'black boxes'. De eerste 'black box' realiseert de functies R(k) en de tweede 'black box' realiseert de functies O(k).
In de ontwerpmethode die in het voorgaande besproken is, is door de keuze van de toestandvariabelen volgens de output-variabelen de tweede 'black box' gedegenereerd tot een aantal doorverbindingen.
Voor volledigheid zal in het kort een alternatieve ontwerp-methode worden besproken welke er op gericht is om de eerste 'black box' om te werken tot doorverbindingen. In dit geval zullen de outputs niet meer afhankelijk zijn van de toestanden en kunnen de outputs worden beschreven als een combinatorische functie van de inputs.

Bij deze ontwerpmethode zullen, indien nodig, secundaire input-variabelen worden gegenereerd waardoor er extra elementen aan het DS worden toegevoegd. De toestanden van deze extra elementen worden zodanig gekozen dat de outputs alleen een functie worden van de nieuwe verzameling inputvariabelen.
Het CS dient er voor te zorgen dat de toestanden van deze extra elementen de juiste waarde hebben. Dit kan bereikt worden door extra instructies (statements) aan de procedure toe te voegen. Bij toevoeging van extra statements kan het CS extra toestanden c.q. extra outputs krijgen. Om na te gaan of de outputs uit de inputvariabelen te realiseren zijn wordt het gehele proces herhaald.
Het nadeel van deze ontwerpmethode is dat de systeembeschrij-ving moet worden aangepast.

Volgens definitie 6.2 geeft P(O) de toestanden van het CS aan die onderscheiden moeten worden. Er bestaat volgens theorema 5.9 een functie om alle toestanden van P(O) te kunnen onder-scheiden als:

$$m\left[\top_H\top E(j), \top_G\top R(i)\right] \leq P(O)$$

ofwel als het triplet:

$$\left[\top_H\top E(j), \top_G\top R(i), P(O)\right]$$

een EPT op het CS is. Dit zijn dus de voorwaarden waaraan de input- en toestandvariabelen moeten voldoen om de toestand van het CS voldoende nauwkeurig te kunnen bepalen.

Het is mogelijk dat het triplet:

$$\left[\overline{\Gamma_j}\, E(j), I, P(0)\right]$$

een EPT op het CS is. Dit betekent dat de toestanden van het CS voldoende nauwkeurig uit de inputs bepaald kunnen worden, maw. de toestanden van het CS zijn combinatorische functies van de inputs en dus kan de eerste 'black box' degenereren tot een aantal doorverbindingen.

Definitie 6.4

Een systeem M is een minimaal systeem dan en slechts dan als het triplet (O,I,Q) een EPT op M is en:

$$Q \leq P(0) \text{ of } m[0,I] \leq P(0)$$

Uit deze definitie blijkt dat T(H,G) voor een minimaal systeem steeds kleiner of gelijk is aan P(O) (H en G beschrijven respectievelijk alle input- en toestandvariabelen)

Voorbeeld 7

Voor het systeem

	a	b	c	d				
1	1	2	-	-	1	0	0	0
2	-	2	3	-	0	1	X	0
3	-	3	3	4	0	X	1	0
4	1	-	-	4	0	0	0	1

geldt $P(0) = \{1;2,3;4\}$
Bij input a gaan toestanden 1/4 over in toestand 1, voor input b in 2,3 , voor input c in 3 en voor input d in 4.
Dus $m(0,I) = \text{Max}\ (1;2,3;3;4) = \{1;2,3;4\}$
Omdat $m(0,I) \leq P(0)$ is het systeem minimaal.

Als het systeem geen minimaal systeem is kunnen alle outputs waarvoor geldt:

$$m[0,I] \leq P(i)$$

waarbij P(i) de EP van de executievoormaarde op S is, als functie van een aantal inputvariabelen beschreven worden. Om de resterende outputs van het CS te kunnen realiseren worden extra

inputvariabelen geïntroduceerd. Deze extra variabelen geven extra EP's. De eisen waaraan deze EP's moeten voldoen worden gegeven door de secundaire voorwaarden.

Definitie 6.5

Stel P(i) is de EP voor een executie voorwaarde op S en Q(i) is een EP op S. Dan is Q(i) de EP voor een secundaire voorwaarde als en slechts als:

$$Q(i).m[0,I] \leq P(i)$$

Theorema 6.3

Stel EP $P(i) = \{ B^1[P(i)] \,.\, B^0[P(i)] \}$ is de EP voor een executievoorwaarde op S. Dan is de EP Q(i):

$$Q(i) = \text{Max}\{ B^1[Q(i)] \,,\, B^0[Q(i)] \}$$

op S de grootst mogelijke EP voor een secundaire voorwaarde als:

$$B^1[Q(i)] = \{ B^1[P(i)] \pm (S - B^0) \}$$
$$B^0[Q(i)] = \{ B^0[P(i)] \pm (S - B^1) \}$$

en:

$$B^1 = \{ \pm [B[F(0)] \,.\, B^1[P(i)]] \mid \text{mits niet } B[F(0)] \leq B^1[P(i)] \}$$
$$B^0 = \{ \pm [B[F(0)] \,.\, B^0[P(i)]] \mid \text{mits niet } B[F(0)] \leq B^0[P(i)] \}$$

Voor het bewijs van dit theorema zie lituratuuropgave 1 hoofdstuk 5.2.

De EP Q(i) kan gebruikt worden om de toestand van de secundaire inputvariabele te bepalen. Dit kan geschieden op omgekeerde wijze als beschreven in theorema 5.3.

Voorbeeld 8

Stel $Q(i) = \{1,2,3;2,3,4\{$. Dan is de toestand van de secundaire inputvariabele gelijk aan X voor toestand 2 en 3, gelijk aan 1 voor toestand 1 en gelijk aan 0 voor toestand 4.

Voor een verzameling $\{Q(i)\}$ kan een verzameling $\{R(j)\}$ worden afgeleid. De verzameling $\{R(j)\}$ is een verzameling 2-bloks EP's zodat iedere R(j) door één variabele gerealiseerd kan worden. Verder moet R(j) zo groot mogelijk zijn zodat de realisatie zo eenvoudig mogelijk is. De EP's R(j) geven de eisen voor de extra inputs.

Definitie 6.6

Stel $\{Q(i)\}$ is de verzameling EP's voor de secundaire voorwaarden. Dan is $\{R(j)\}$ de verzameling 2 bloks EP's die alle Q(i) realiseren als en slechts als:

1. Iedere R(j) 2 blokken bevat
2. $R(k) \leq R(j)$ impliceert j=k
3. Voor iedere Q(i) is er een R(j) zodanig dat:

$$R(j) \leq Q(i)$$

De verzameling $\{R(j)\}$ kan berekend worden door het product van alle paren $\{Q(i)\}$ en alle product paren te bepalen. Het product vervangt het paar als dit een EP met twee blokken is, of een EP met twee blokken die van het product is afgeleid.

De EP's R(j) geven de eisen voor de extra inputs van het CS. Hieruit kunnen op dezelfde manier als in voorbeeld 8 de toestanden van de extra inputvariabelen worden afgeleid.

De eisen van de extra inputvariabelen worden aan de ontwerper gepresenteerd. Deze kan extra elementen aan het DS toevoegen. Omdat de procedure geen combinatorische netwerk kan beschrijven dient het extra element een flipflop te zijn. Is dit een set-reset flipflop dan vormen de set- en resetsignalen van de flipflop outputs van het CS. De output van de flipflop is tevens input van het CS.
Hoe de beschrijving van het CS moet worden gewijzigd kan, als de toestanden van de extra inputvariabelen bekend zijn, gemakkelijk worden afgeleid met de volgende vuistregel:
Er moet een extra statement worden gecreëerd bij 1-0 resp. 0-1 overgangen; de instructie kan in statement S worden opgenomen bij X-0 resp. X-1 overgangen.

Voorbeeld 9

Stel dat de toestanden voor een extra element worden gegeven door de EP:

$$R = \{1,3,4,5;1,2,5\}$$

De toestanden van de extra inputvariabele zijn dan:

	1	2	3	4	5
r	X	0	1	1	X

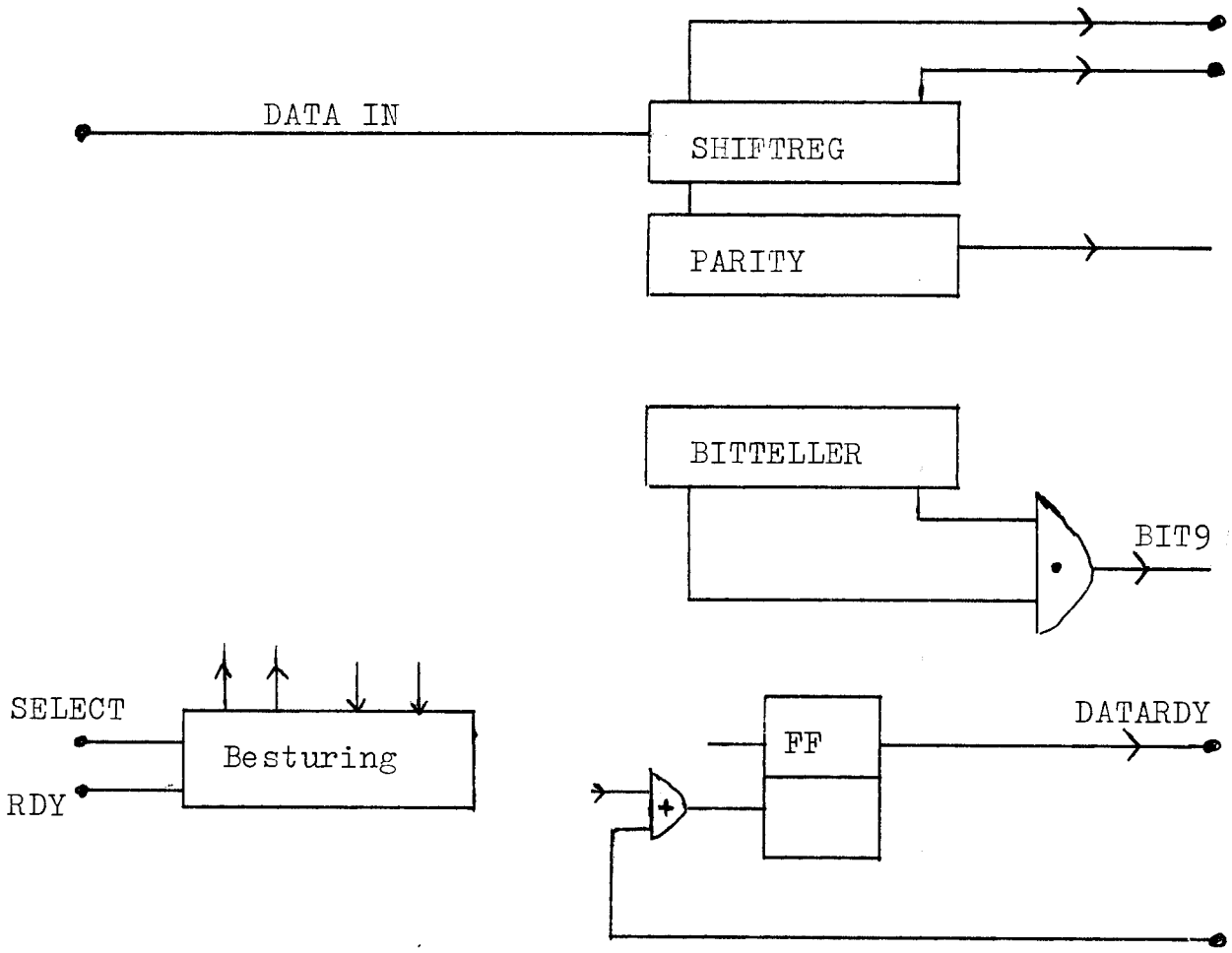
Tussen statement 2 en statement 3 dient een set-instructie te worden aangebracht. In statement 5 of statement 1 kan een reset-instructie worden opgenomen.

Van de nieuwe systeembeschrijving wordt nagegaan of een minimaal systeem wordt beschreven. Is dit niet het geval dan worden weer extra inputvariabelen gegenereerd,enzovoorts. Het proces is eindig omdat het DS maximaal het gehele systeem M kan beschrijven. Het CS is dan gereduceerd tot een systeem met 1 toestand, 1 input en 1 output. Dit is een minimaal systeem.

De outputs van een minimaal systeem zijn gemakkelijk af te leiden uit de bij de simulatie gegenereerde tabel. Omdat de outputs voor een minimaal systeem niet meer afhangen van de toestanden kunnen nu zowel de kolom d(s,x) als de kolom s worden genegeerd. Verder kunnen de outputfuncties uit de functietabel die ontstaat, worden afgeleid op de reeds besproken wijze.

Bij de beschreven ontwerpprocessen werden een aantal functies ontwikkeld die het te ontwerpen systeem beschreven. Deze functies dienen nu te worden omgezet in declaraties. Daarna kan het ontworpen systeem worden gesimuleerd zodat het werkelijke gedrag van het systeem kan worden bepaald d.w.z. het gedrag rekening houdend met vertragingen en andere niet ideale mogelijkheden. Daarna zal het systeem worden gedetailleerd voor de bouw en kan de gedetailleerde systeembeschrijving worden gegeven. Tenslotte kan het gebouwde systeem (automatisch) worden getest.
De bespreking van deze ontwerpstappen valt buiten het bestek van deze inleiding tot het automatisch ontwerpen van digitale systemen.

7. VOORBEELD TER ILLUSTRATIE VAN HET AUTOMATISCH ONTWERPEN VAN DIGITALE SYSTEMEN GEBRUIK MAKEND VAN DSL2



figuur 7.1

In dit hoofdstuk zal de besturing worden ontworpen van een digitaal systeem dat het schuiven van informatie in een schuifregister beschrijft. De DSL2-beschrijving voor dit systeem is reeds gegeven in hoofdstuk 4.3 voorbeeld 2. Voordat een digitaal systeem wordt ontworpen zijn reeds een aantal elementen van het systeem bekend. In figuur 7.1 zijn de elementen voor dit voorbeeld aangegeven. Deze elementen komen allen voor in de declaratie statements 16 t/m 22 van voornoemde systeembeschrijving.
Wat het systeem moet doen wordt beschreven in de procedure. De procedure beschrijft de besturing welke moet worden ontworpen.

Door representatieve simulatie kan de toestand sequentie tabel en de output tabel worden bepaald. Het resultaat van deze simulatie is reeds gegeven in tabel 4.2.2 van hoofdstuk 4.3. Uit deze tabel kunnen de volgende P(k) worden afgeleid:

$P(4) = \{4,6;6,8,9,10,12,13,14\}$

$P(9) = \{9,10;4,6,8,10,12,13,14\}$

$P(12) = \{12;4,6,8,9,10,12,14\}$

Codering van de toestanden volgens de output variabelen geeft:

4: 100
6: X00
8: 000
9: 010
10: 0X0
12: 001
13: 000
14: 000

Bepaling van T(H,G) voor alle input variabelen en toestand variabelen geeft:

$T(H,G) = \{4,6;8;9,10;12,14;13,14\}$

zodat :

$T(H,G) \leq P(4)$

$T(H,G) \leq P(9)$

P(12) wordt niet gerealiseerd.

Ter bepaling van de secundaire toestand variabelen wordt P(0) berekend:

$P(0) = \{4,6;6,8,10,13,14;9,10;12\}$

Voor de bepaling van Q wordt eerst $B(\bar{x},j,i)$ bepaald volgens:

$B(x,j,i) = \{ S-\{s\} \mid d(s,x) \notin Bpo(i)$ als S in $Bpo(j)$

waarbij Bpo een blok van P(0) voorstelt.

Bpo(j)	B(x,j,i)
{4,6}	o.a. {4,6,8,9,10,12,13,14}
{6,8,10,13,14}	{4,6,8,9,10,12} als Bpo(i) = {6,8,10,13,14}
	{4,6,8,9,12,13,14} als Bpo(i) = {12}
{9,10}	o.a. {4,6,8,9,10,12,13,14}
{12}	o.a. {4,6,8,9,10,12,13,14}

Dan is $Q = \bigcap_{x,j} \{\max_i B(x,j,i)\} = \{4,6,8,9,10,12;4,6,8,9,12,13,14\}$

Dit is een 2-bloks EP. Deze wordt als R(4) aan de outputs toegevoegd.

Opnieuw bepaling van T(H,G) voor alle variabelen levert:

$$T(H,G) \leq P(0)$$

We vinden nu de volgende functie-tabel:

	SEL	RDY	BIT9	PAR	DAT	4	9	12	R(4)	4	9	12	R(4)
1	0	X	X	X	X	1	0	0	X	1	0	0	X
2	0	0	1	X	0	0	0	0	0	1	0	0	X
3	0	0	1	X	0	0	0	0	0	1	0	0	X
4	0	X	0	X	0	1	0	0	X	X	0	0	X
5	0	X	0	X	0	X	0	0	X	X	0	0	X
6	1	0	0	X	0	X	0	0	X	0	0	0	X
7	1	0	0	X	0	0	0	0	X	0	0	0	X
8	1	0	0	X	0	0	X	0	1	0	0	0	X
9	1	1	0	X	0	0	0	0	X	0	1	0	X
10	1	1	0	X	0	0	1	0	X	0	X	0	1
11	1	1	0	X	0	0	X	0	1	0	X	0	1
12	1	0	1	0	0	0	X	0	1	0	0	1	X
13	1	0	1	0	0	0	0	1	X	0	0	1	X
14	1	0	1	1	0	0	X	0	1	0	0	0	0
15	1	0	1	0	1	0	0	1	X	0	0	0	0
16	1	0	1	0	1	0	0	0	0	0	0	0	0
17	1	0	1	X	0	0	0	0	0	0	0	0	0
18	1	0	1	X	0	0	0	0	0	0	0	0	0

Uit de tabel volgt voor de outputs:

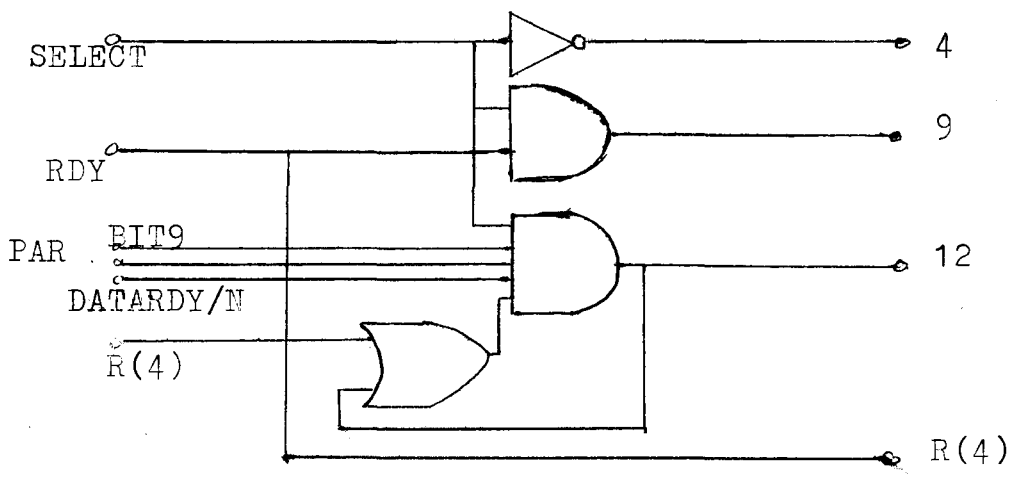
4 = B(0) OF SELECT

9 = B(11) OF SELECT, RDY

12 = B(110001,11001X) OF SELECT,BIT9,PARITY,DATARDY,12,R(4).

R(4)= B(1) OF RDY

In figuur 7.2 is de ontworpen schakeling gegeven.



Figuur 7.2: Schakeling van de besturing in figuur 7.1.

8. CONCLUSIES

Uit de lituratuurstudie bleek, dat DSL2 de enigste ontwerptaal was die werd ontwikkeld vanuit de definitie van een digitaal systeem en die onafhankelijke procedures kende. De systeembeschrijving in DSL2 leende zich goed voor omzetting in een toestand sequentie-tabel en output-tabel. Van alle andere genoemde talen gaat de autòmatisering niet verder dan simulatie en omzetting van de beschrijving in Boolse functies. In het DSL2-systeem is het ontwerpen geautomatiseerd tot en met de bouw en testen van het systeem toe.
Het gebruik van Extended Partition Triplets bleek een zeer geschikte manier om de informatie-flow van het subsysteem,beschreven,in het Control System,te beschrijven. Hierbij bleken de MMm EPT's alle informatie te geven welke nodig was voor het toewijzen van variabelen aan de toestand van de machine. Het bepalen van de MMm paren kan geschieden door een computerprogramma. Hierbij moet men zich echter beperken tot systemen met een klein aantal inputs en toestanden.
Er werden twee methodes beschreven om de beschrijving van procedures om te zetten in Boolse functies. Hierbij verdient de methode waarbij extra toestandvariabelen moeten worden geintroduceerd de voorkeur omdat voor deze methode de systeembeschrijving niet aangepast hoeft te worden. De methode garandeert geen eenduidige oplossing maar wel worden alle zinvolle mogelijkheden gegenereerd welke aan de gestelde beperkingen voldoen. De ontwerper kan hieruit een oplossing kiezen.
De gekozen oplossing geeft geen garantie voor het vinden van het 'beste' ontwerp. Hiertoe zullen alle MMm-EPT's moeten worden bepaald en alle mogelijke oplossingen moeten worden nagelopen.
Voor beide ontwerpmethodes kon de toestand-sequentie tabel tot een normale functietabel worden omgezet zodat de Boolse functies voor de outputs van het Control System gemakkelijk konden worden afgeleid.

Voor de goede orde zij vermeld dat ten tijde van de samenstelling van dit verslag de compiler voor de taal DSL2 practisch is voltooid en dat voor alle ontwerpstappen reeds programma' s zijn ontwikkeld. Een uitzondering hierop vormt het programma wat de toestand-sequentie tabel uit de procedure moet genereren. Verder zijn de verschillende programma's nog niet aan elkaar aangepast.

9. Lituratuurlijst

(1). J.Hoogeveen, "AUTOMATISCH ONTWERPEN van Digitale Systemen", Intern rapport Techn.Hogeschool Eindhoven,Groep ECB,mei 1972.

(2). G.Gordon, " A general purpose systems simulation program," 1961 Proc. EJCC, pp. 87-104.

(3). D.E.Knuth and J.L. McNeley, " SOL-A symbolic language for general-purpose systems simulation," IEEE Trans. on Electronic Computers, vol. EC-13, pp. 401-408, August 1964.

(4). D.E.Knuth and J.L.McNeley," A formal definition of SOL, " IEEE Trans. on Electronic Computers, vol. EC-13, pp. 409-414, August 1964.

(5). M.I. Youchah, et al., " The data processing systems simulator (DPSS), " Systems development Corp., Paramus, N.J., Tech. Rept. SP-1299/000/01.

(6). M.V.Wilkes and J.B.Stringer, " Micro-programming and the design of the control circuits in an electronic digital computer," Proc.Cambridge Phil. Socs., vol. 44, pt 2, pp. 230-238, April 1953.

(7). I.S. Reed, Symbolic synthesis of digital computers," Proc. ACM, pp. 90-94, September 8-10, 1952.

(8). Y.Chu, Digital Computer Design Fundamentals. New York: McGraw-Hill, 1962.

(9). H.Schorr, " Computer-aided digital systems design and analysis using a register transfer language," IEEE Trans. Electronic Computers, vol. EC-13, pp. 730-737, December 1964.

(10).D.F.Gorman and J.P.Anderson," A logic design translator, " 1962 Fall Joint Computer Conf., AFIPS Proc., vol.22. Washington, D.C.: Spartan, 1962, pp. 251-261.

(11).R.M.Proctor, " A logic design translator experiment demonstrating relationships of language to systems and logic design," IEEE Trans. Electronic Computers, vol. EC-13, pp. 422-430, August 1964.

(12).G.Estrin and R.Mandell, " Metacompiler as a design automation tool," 1966 Proc. Share Design Automation Conference.

(13). J.P.Roth, " Systematic design of automata" 1965 Fall Joint Computer Conf., AFIPS Proc., vol. 27, pt 1. Washington, D.C.: Spartan, 1965, pp. 1093-1100.

(14). K.E.Iverson, A Programming Language. New York: Wiley, 1962.

(15). K.E.Iverson, " A common language for hardware, software and applications, " 1962 Fall Joint Computer Conf., AFIPS Proc., vol. 22. Washington, D.C.:Spartan, 1962, pp. 121-129.

(16). K.E.Iverson," A programming notation in system design," IBM Sys.J., vol. 2, pp. 117-128, June 1963.

(17). A.D.Falkoff, K.E.Iverson, and E.H.Sussenguth, " A formal description of system/360," IBM Sys.J., vol. 3, no. 3, pp. 198-262, 1964.

(18). H.P. Schlaeppi, "A formal language for describing machine logic, timing and sequencing (LOTIS)," IEEE Trans. Electronic Computers, vol. EC-13, p.p. 439-448, August 1964.

(19). D.L. Parnas, "A language for describing the functions of synchronous systems", Commun. ACM, vol.9, pp.72-76, February 1966.

(20). Y. Chu, "An algol-like computer design language", Commun. ACM, vol.8, pp. 607-615, October 1965.

(21). E.P. Stabler, "System Description Languages", IEEE Trans. on Computers, vol. C-19, no. 12, december 1970, p.p. 1160-1173.

(22). E.D. Crockett et al., "Computer-aided system design", 1970 Fall Joint Computer Conf.,AFIPS Proc., vol. 37, pp. 286-296.

(23). G. Metze, S. Seshu, "A proposal for a computer compiler", 1966 Spring Joint Computer Conf., AFIPS Proc., vol. 28, Washington, D.C.: Spartan, 1966, pp. 253-264.

(24). J.R. Duley, D.L. Dietmeyer, " A Digital System Design Language (DDL)", IEEE Trans. on Computers, vol. C-17, no 9, september 1968, p.p.850-861.

(25). J. R. Duley, D.L. Dietmeyer, "Translation of a DDL Digital System Specification to Boolean Equations", IEEE Trans. on Computers, vol. C-18, no. 4, p p. 305-313, april 1969.

(26). G.B.Gerace, " Digital System Design Automation - A Method for Designing a Digital System as a Sequential Network System," IEEE Trans. on Computers, vol. C-17, no. 11, pp. 1044-1061, November 1968.

(27). G.Bogo et al.,"CASSANDRE and the Computer Aided Logical Systems Design", IFIP Congress 71, Booklet TA - 6, pp. 26 - 32, Ljubljana- August 1971.

(28). M.A.Breuer," General Survey of Design Automation of Digital Computers", Proc. of the IEEE, vol. 54, no. 12, pp. 1708-1721, December 1966.

(29). J.Hartmanis and R.E.Stearns, " Algebraic Structure Theory of Sequential Machines" Prentice-Hall, Inc. Englewood Cliffs, N.J. 1966.