Contribution à la conception assistée par calculateur des systèmes logiques

P. Deschizeaux

N° d'ordre :

THÈSE

présentée

A L'UNIVERSITÉ SCIENTIFIQUE ET MÉDICALE DE GRENOBLE

pour obtenir

LE TITRE DE DOCTEUR ÈS-SCIENCES PHYSIQUES

PAR

Pierre DESCHIZEAUX

Contribution à la conception assistée par calculateur des systèmes logiques.

Thèse soutenue le 12 Novembre 1971 devant la Commission d'Examen

JURY

MM. L. NÉEL — *Président*

J. KUNTZMANN
C. DURANTE
R. PERRET
— *Examinateurs*

UNIVERSITE SCIENTIFIQUE
ET MEDICALE DE GRENOBLE

LISTE DES PROFESSEURS

Président : Monsieur Michel SOUTIF
Vice-Président : Monsieur Gabriel CAU

PROFESSEURS TITULAIRES

MM.	ANGLES D'AURIAC Paul	Mécanique des fluides
	ARNAUD Georges	Clinique des maladies infectieuses
	ARNAUD Paul	Chimie
	AYANT Yves	Physique approfondie
	BARBIER Jean-Claude	Physique expérimentale
	BARBIER Reynold	Géologie appliquée
	BARJON Robert	Physique nucléaire
	BARNOUD Fernand	Biosynthèse de la cellulose
	BARRIE Joseph	Clinique chirurgicale
	BENOIT Jean	Radioélectricité
	BESSON Jean	Electrochimie
	BEZES Henri	Chirurgie générale
	BLAMBERT Maurice	Mathématiques pures
	BOLLIET Louis	Informatique (IUT B)
	BONNET Georges	Electrotechnique
	BONNET Jean-Louis	Clinique ophtalmologique
	BONNET-EYMARD Joseph	Pathologie médicale
	BONNIER Etienne	Electrochimie Electrométallurgie
	BOUCHERLE André	Chimie et Toxicologie
	BOUCHEZ Robert	Physique nucléaire
	BRAVARD Yves	Géographie
	BRISSONNEAU Pierre	Physique du Solide
	BUYLE-BODIN Maurice	Electronique
	CABANAC Jean	Pathologie chirurgicale
	CABANEL Guy	Clinique rhumatologique et hydrologique
	CALAS François	Anatomie
	CARRAZ Gilbert	Biologie animale et pharmacodynamie
	CAU Gabriel	Médecine légale et Toxicologie
	CAUQUIS Georges	Chimie organique
	CHABAUTY Claude	Mathématiques pures
	CHATEAU Robert	Thérapeutique
	CHENE Marcel	Chimie papetière
	COEUR André	Pharmacie chimique
	CONTAMIN Robert	Clinique gynécologique
	COUDERC Pierre	Anatomie Pathologique
	CRAYA Antoine	Mécanique
Mme	DEBELMAS Anne-Marie	Matière médicale
MM.	DEBELMAS Jacques	Géologie générale
	DEGRANGE Charles	Zoologie
	DESSAUX Georges	Physiologie animale
	DODU Jacques	Mécanique appliquée
	DREYFUS Bernard	Thermodynamique
	DUCROS Pierre	Cristallographie
	DUGOIS Pierre	Clinique de Dermatologie et Syphiligraphie

	FAU René	Clinique neuro-psychiatrique
	FELICI Noël	Electrostatique
	GAGNAIRE Didier	Chimie physique
	GALLISSOT François	Mathématiques pures
	GALVANI Octave	Mathématiques pures
	GASTINEL Noël	Analyse numérique
	GERBER Robert	Mathématiques pures
	GIRAUD Pierre	Géologie
	KLEIN Joseph	Mathématiques pures
Mme	KOFLER Lucie	Botanique et physiologie végétale
MM.	KOSZUL Jean-Louis	Mathématiques pures
	KRAVTCHENKO Julien	Mécanique
	KUNTZMANN Jean	Mathématiques appliquées
	LACAZE Albert	Thermodynamique
	LACHARME Jean	Biologie végétale
	LATURAZE Jean	Biochimie pharmaceutique
	LEDRU Jean	Clinique médicale B
	LLIBOUTRY Louis	Géophysique
	LOUP Jean	Géographie
Mle	LUTZ Elisabeth	Mathématiques pures
MM.	MALGRANGE Bernard	Mathématiques pures
	MALINAS Yves	Clinique obstétricale
	MARTIN-NOEL Pierre	Séméiologie médicale
	MASSEPORT Jean	Géographie
	MAZARE Yves	Clinique médicale A
	MICHEL Robert	Minéralogie et Pétrographie
	MOURIQUAND Claude	Histologie
	MOUSSA André	Chimie nucléaire
	NEEL Louis	Physique du Solide
	OZENDA Paul	Botanique
	PAUTHENET René	Electrotechnique
	PAYAN Jean-Jacques	Mathématiques pures
	PEBAY-PEYROULA Jean-Claude	Physique
	PERRET René	Servomécanismes
	PILLET Emile	Physique industrielle
	RASSAT André	Chimie systématique
	RENARD Michel	Thermodynamique
	REULOS René	Physique industrielle
	RINALDI Renaud	Physique
	ROGET Jean	Clinique de pédiatrie et de puériculture
	SANTON Lucien	Mécanique
	SEIGNEURIN Raymond	Microbiologie et Hygiène
	SENGEL Philippe	Zoologie
	SILBERT Robert	Mécanique des fluides
	SOUTIF Michel	Physique générale
	TANCHE Maurice	Physiologie
	TRAYNARD Philippe	Chimie générale
	VAILLAND François	Zoologie
	VAUQUOIS Bernard	Calcul électronique
Mme	VERAIN Alice	Pharmacie galénique
	VERAIN André	Physique
Mme	VEYRET Germaine	Géographie
MM.	VEYRET Paul	Géographie
	VIGNAIS Pierre	Biochimie médicale
	YOCCOZ Jean	Physique nucléaire théorique

	BEGUIN Claude	Chimie organique
	BELORIZKY Elie	Physique
	BENZAKEN Claude	Mathématiques appliquées
Mme	BERTRANDIAS Françoise	Mathématiques pures
MM.	BLIMAN Samuel	Electronique (EIE)
	BLOCH Daniel	Electrotechnique
Mme	BOUCHE Liane	Mathématiques (CUS)
MM.	BOUCHET Yves	Anatomie
	BOUSSARD Jean-Claude	Mathématiques appliquées
	BOUVARD Maurice	Mécanique des Fluides
	BRIERE Georges	Physique expérimentale
	BRODEAU François	Mathématiques (IUT B)
	BRUGEL Lucien	Energétique
	BUISSON Roger	Physique
	BUTEL Jean	Orthopédie
	CHAMBAZ Edmond	Biochimie médicale
	CHAMPETIER Jean	Anatomie et organogénèse
	CHARACHON Robert	Oto-Rhino-Laryngologie
	CHIAVERINA Jean	Biologie appliquée (EFP)
	CHIBON Pierre	Biologie animale
	COHEN-ADDAD Jean-Pierre	Spectrométrie physique
	COLOMB Maurice	Biochimie médicale
	CONTE René	Physique
	CROUZET Guy	Radiologie
	DURAND Francis	Métallurgie
	DUSSAUD René	Mathématiques (CUS)
Mme	ETERRADOSSI Jacqueline	Physiologie
MM.	FAURE Jacques	Médecine légale
	GAVEND Michel	Pharmacologie
	GENSAC Pierre	Botanique
	GERMAIN Jean-Pierre	Mécanique
	GIDON Maurice	Géologie
	GRIFFITHS Michael	Mathématiques appliquées
	GROULADE Joseph	Biochimie médicale
	HOLLARD Daniel	Hématologie
	HUGONOT Robert	Hygiène et médecine préventive
	IDELMAN Simon	Physiologie animale
	IVANES Marcel	Electricité
	JALBERT Pierre	Histologie
	JOLY Jean-René	Mathématiques pures
	JOUBERT Jean-Claude	Physique du Solide
	JULLIEN Pierre	Mathématiques pures
	KAHANE André	Physique générale
	KUHN Gérard	Physique
Mme	LAJZEROWICZ Jeannine	Physique
MM.	LAJZEROWICZ Joseph	Physique
	LANCIA Roland	Physique atomique
	LE JUNTER Noël	Electronique
	LEROY Philippe	Mathématiques
	LOISEAUX Jean-Marie	Physique nucléaire
	LONGEQUEUE Jean-Pierre	Physique nucléaire
	LUU DUC Cuong	Chimie organique
	MACHE Régis	Physiologie végétale
	MAGNIN Robert	Hygiène et Médecine préventive
	MARECHAL Jean	Mécanique
	MARTIN-BOUYER Michel	Chimie (CUS)
	MAYNARD Roger	Physique du Solide
	MICOUD Max	Maladies infectieuses
	MOREAU René	Hydraulique (INP)

PROFESSEURS ASSOCIES

MM.	BULLEMER Bernhard	Physique
	RADHAKRISHNA Pidatala	Thermodynamique

PROFESSEURS SANS CHAIRE

MM.	AUBERT Guy	Physique
Mme	BARBIER Marie-Jeanne	Electrochimie
MM.	BARRA Jean	Mathématiques appliquées
	BEAUDOING André	Pédiatrie
	BERTRANDIAS Jean-Paul	Mathématiques appliquées
	BIAREZ Jean-Pierre	Mécanique
	BONNETAIN Lucien	Chimie minérale
Mme	BONNIER Jane	Chimie générale
MM.	CARLIER Georges	Biologie végétale
	COHEN Joseph	Electrotechnique
	COUMES André	Radioélectricité
	DEPASSEL Roger	Mécanique des Fluides
	DEPORTES Charles	Chimie minérale
	DESRE Pierre	Métallurgie
	DOLIQUE Jean-Michel	Physique des plasmas
	GAUTHIER Yves	Sciences biologiques
	GEINDRE Michel	Electroradiologie
	GIDON Paul	Géologie et Minéralogie
	GLENAT René	Chimie organique
	HACQUES Gérard	Calcul numérique
	JANIN Bernard	Géographie
Mme	KAHANE Josette	Physique
MM.	LATREILLE René	Chirurgie générale
	LAURENT Pierre	Mathématiques appliquées
	MULLER Jean-Michel	Thérapeutique
	PERRIAUX Jean-Jacques	Géologie et minéralogie
	POULOUJADOFF Michel	Electrotechnique
	REBECQ Jacques	Biologie (CUS)
	REVOL Michel	Urologie
	REYMOND Jean-Charles	Chirurgie générale
	ROBERT André	Chimie papetière
	SARRAZIN Roger	Anatomie et chirurgie
	SARROT-REYNAULD Jean	Géologie
	SIBILLE Robert	Construction Mécanique
	SIROT Louis	Chirurgie générale
Mme	SOUTIF Jeanne	Physique générale
M.	VALENTIN Jacques	Physique nucléaire

MAITRES DE CONFERENCES ET MAITRES DE CONFERENCES AGREGES

Mle	AGNIUS-DELORD Claudine	Physique pharmaceutique
	ALARY Josette	Chimie analytique
MM.	AMBLARD Pierre	Dermatologie
	AMBROISE-THOMAS Pierre	Parasitologie
	ARMAND Yves	Chimie

	NEGRE Robert	Mécanique
	PARAMELLE Bernard	Pneumologie
	PECCOUD François	Analyse (IUT B)
	PEFFEN René	Métallurgie
	PELMONT Jean	Physiologie animale
	PERRET Jean	Neurologie
	PERRIN Louis	Pathologie expérimentale
	PFISTER Jean-Claude	Physique du Solide
	PHELIP Xavier	Rhumatologie
Mle	PIERY Yvette	Biologie animale
	RACHAIL Michel	Médecine interne
	RACINET Claude	Gynécologie et obstétrique
	RICHARD Lucien	Botanique
Mme	RINAUDO Marguerite	Chimie macromoléculaire
MM.	ROMIER Guy	Mathématiques (IUT B)
	ROUGEMONT (DE) Jacques	Neuro-chirurgie
	STIEGLITZ Paul	Anesthésiologie
	STOEBNER Pierre	Anatomie pathologique
	VAN CUTSEM Bernard	Mathématiques appliquées
	VEILLON Gérard	Mathématiques appliquées (INP)
	VIALON Pierre	Géologie
	VOOG Robert	Médecine interne
	VROUSSOS Constantin	Radiologie
	ZADWORNY François	Electronique

MAITRES DE CONFERENCES ASSOCIES

MM.	BOUDOURIS Georges	Radioélectricité
	CHEEKE John	Thermodynamique
	GOLDSCHMIDT Hubert	Mathématiques
	YACOUD Mahmoud	Médecine légale

CHARGES DE FONCTIONS DE MAITRES DE CONFERENCES

Mme	BERIEL Hélène	Physiologie
Mme	RENAUDET Jacqueline	Microbiologie

Je tiens à assurer de ma profonde reconnaissance Monsieur le Professeur NEEL, Prix Nobel de Physique, Membre de l'Institut, Directeur de l'Institut National Polytechnique de Grenoble, qui m'a fait l'honneur d'accepter la présidence du jury.

Je remercie vivement Monsieur le Professeur KUNTZMANN qui, après m'avoir formé à la recherche dans son laboratoire, a regardé avec intérêt mon travail et a accepté de faire partie du jury.

J'adresse tous mes remerciements à mon parrain au C. N. R. S., Monsieur le Professeur DURANTE, qui s'est intéressé à mon travail et a accepté de participer au jury.

Je remercie Monsieur le Professeur PERRET qui m'a accueilli dans son laboratoire et dont les conseils et encouragements m'ont été infiniment précieux.

Enfin, je remercie tous mes camarades du laboratoire d'Automatique et particulièrement R. DAVID, dont les nombreuses suggestions m'ont beaucoup aidé.

C H A P I T R E 1

INTRODUCTION

Notre travail est une contribution aux nombreuses recherches effectuées dans le domaine de la conception assistée des systèmes logiques. Il a suivi l'évolution générale des travaux analogues : d'un aspect très restreint d'aide au calcul algébrique, il est passé à l'étude de méthodes très complètes d'aide à la conception.

Il y a quelques années, en effet, le problème de la conception assistée était très différent. Tout d'abord, la taille des systèmes à concevoir était faible, car la fiabilité des composants, leur encombrement et leur échauffement interdisaient les trop gros ensembles. Par ailleurs, la technologie était moins élaborée que maintenant, ce qui avait l'heureux effet d'imposer des contraintes commodes. Par exemple, quand les opérateurs ET et OU étaient réalisés avec des diodes, il était en pratique, impossible de disposer, sans régénération, plus de deux opérateurs en série ; cela imposait donc de réaliser les fonctions booléennes sous forme polynomiale, ce qui, mathématiquement, est très commode. Un autre exemple se trouve dans les systèmes séquentiels : quand les éléments combinatoires se faisaient avec des diodes peu coûteuses et les bascules avec des transistors nettement plus chers, il était suffisant pour obtenir un coût total faible, de minimiser le nombre de variables internes et donc de bascules. Enfin, les moyens de calcul dont on disposait ne permettaient pas des programmes complexes. Il était possible d'écrire des algorithmes algébriques simples ne faisant pas intervenir toutes les astuces employées par l'homme ; un exemple typique des programmes d'alors est le calcul des composants premiers d'une fonction par la méthode de Quine et Mc Cluskey qui a pu être mise en oeuvre sur très petits calculateurs.

La deuxième raison est psychologique ! Les réseaux calculés par un ordinateur semblent à première vue bizarres, et, puisque c'est un homme qui doit assurer la maintenance des circuits et leur dépannage, il est souhaitable que ce soit un homme qui les conçoive.

Actuellement, une troisième révolution technologique va mettre tout le monde d'accord, logiciens et fabricants : les circuits intégrés à grande échelle bouleversent les traditions. Une caractéristique essentielle de ces circuits est de n'être ni dépannables, ni modifiables.

Ils ne sont pas dépannables, donc la contrainte de simplicité disparait : puisque personne ne touche aux circuits, il est sans importance que personne ne les comprenne.

Ils ne sont pas modifiables, donc toute erreur de conception devient catastrophique. En général, on admet en effet que le prix de revient d'une série est pratiquement celui des photomasques et toute erreur dans ceux-ci équivaut à la mise au rebut de toute la série.

Ce dernier aspect rend indispensable l'emploi de l'ordinateur qui sera utilisé non plus pour faire mieux que l'homme, mais pour faire, sans se tromper, la même chose.

Une autre raison d'utiliser l'ordinateur est le regain d'intérêt de la minimisation. Avec les composants discrets, le nombre d'éléments n'avait

- 3 - implantation et cablage. Le schéma logique est transformé en un schéma réel, compte tenu de la technologie utilisée, de la manière de disposer les éléments sur un support matériel, de la manière de réaliser les connexions, etc.....

- 4 - vérification du réseau. Cette opération est indispensable quand les étapes précédentes ont été faites manuellement ; elle consiste en général en une simulation, soit logique et grossière, soit plus fine en tenant compte des propagations, des temps de transitions etc....

- 5 - définition d'un test. Cette opération consiste à définir dans quelles conditions doit s'effectuer le test du réseau pour déceler et éventuellement localiser les pannes.

- 6 - fabrication. Dans certains cas on peut envisager une fabrication assistée : par exemple le cablage automatique de plaquettes, ou le dessin automatique des masques de circuits intégrés.

Notre travail se situe essentiellement au niveau de la synthèse logique, bien que nous ayons tenu compte de certains problèmes d'implantation et de test. Il a trois aspects : mathématique, physique et informatique.
Dans le domaine mathématique, nous avons étudié des méthodes générales de calcul des réseaux combinatoires et des réseaux séquentiels asynchrones.
Dans le domaine physique nous avons étudié les problèmes électroniques de l'intégration à grande échelle. Cette étude qui n'apparaîtra pratiquement pas ici, nous a permis de découvrir les contraintes pratiques auxquelles sont soumis les fabricants de circuits, et les facteurs d'évaluation du coût des circuits.
Enfin, dans le domaine informatique, nous avons écrit de nombreux programmes d'aide à la conception, dans une première étape afin de comparer

les méthodes théoriques, et dans une deuxième pour développer un outil effectivement utilisable.

Une des originalités de notre travail est l'étroite imbrication entre la partie mathématique et la partie informatique : c'est en écrivant des programmes relatifs à des méthodes connues que nous avons pu voir leur faiblesse, et nous orienter vers des méthodes entièrement nouvelles. Toutes les théories que nous présentons dans cette thèse ont pour origines les contraintes de la programmation. Cet aspect s'est avéré très fécond, et des généralisations qui, mathématiquement n'étaient pas évidentes, ont été découvertes intuitivement en cherchant à programmer astucieusement des méthodes déjà connues. Par ailleurs certaines études statistiques sur les réseaux obtenus par programmes ont permis de comparer certaines méthodes et d'orienter nos recherches.

Synthèse des réseaux combinatoires.

Ce sujet, abordé depuis longtemps par de nombreux théoriciens, a été traité ici par des méthodes nouvelles. L'idée générale de ces méthodes est de "fractionner" chaque fonction à réaliser en sous-fonctions de plus en plus simples, en employant donc un procédé récursif qui utilise au mieux les possibilités des langages modernes de programmation. Suivant la manière dont se fait la décomposition, on peut obtenir des méthodes très diverses.

- 1 - Une première méthode, adaptée à une synthèse à l'aide d'opérateurs NOR par exemple, consiste à fractionner chaque fonction F en K fonctions $F_1, F_2 \ldots F_k$ vérifiant : $F = NOR(F_1, F_2 \ldots F_k)$
On obtient les F_i par une partition des monômes d'une base irredondante

de F_1. Cette méthode que nous avons appelé méthode de "fractionnement littérale" utilise la théorie habituelle de minimisation des formes polynomiales. Bien que les programmes relatifs à cette méthode soient lourds (relativement à ceux qui suivront), elle est intéressante pour trois raisons.

a) On obtient des réseaux de structure simple et facilement compréhensible.

b) Elle permet de traiter le problème des aléas en cours de calcul.

c) Elle a permis enfin d'aborder le problème des tests de manière très simple.

Nous l'avons programmée depuis plusieurs années et en avons fait le sujet d'une thèse de docteur-ingénieur en 1967.

- 2 - Une deuxième méthode est née de la lenteur relative de la première : les ordinateurs ne sont pas faits pour le calcul littéral et celui-ci est donc lent. Nous avons préféré alors une méthode qui utilise une <u>représentation vectorielle</u> des fonctions booléennes et des notions de <u>distances</u>, beaucoup plus adaptées aux possibilités du calcul automatique. Par ailleurs, nous remarquions que, dans les méthodes habituelles, les réseaux obtenus sont arborescents et donc coûteux, ce qui s'explique intuitivement en disant que la probabilité de trouver dans un réseau deux sous-fonctions F_i et F_j égales, est très faible. Nous avons alors cherché une méthode qui permette de choisir <u>a priori</u> ce qui, dans une partie de réseau déjà existant, resservira dans une autre partie.

L'idée de base de notre méthode se dégage alors natu ellement.

Désirant réaliser F et disposant de fonctions déjà réalisées F_1, F_2.. .....F_p, on cherchera a priori à utiliser celle qui se "rapproche" le plus de F, quitte à la corriger en lui "enlevant les points en trop" et lui rajoutant les "points qui manquent". Cette méthode fait donc intervenir une <u>distance</u> entre fonctions et nous l'avons appelée

méthode topologique. Elle est plus facile à programmer, plus rapide et donne des résultats plus économiques que la précédente. En outre elle a l'immense avantage de permettre de traiter simultanément synthèse, implantation et cablage. En effet, on voit que cherchant la meilleure fonction F_i pour réaliser F, on peut également chercher celle qui est la plus proche dans le réseau du point où est réalisé F ; on cherche celle qui nécessitera une connexion facile (compte tenu des croisements de connexions par exemple).

Cette méthode est destinée à l'intégration à grande échelle ; nous avons pu la programmer en faisant des hypothèses raisonnables sur le cablage, sans toutefois avoir pu la mettre réellement en pratique faute de problèmes précis.

Synthèse séquentielle.

La deuxième partie de cette thèse est consacrée à la mise en application et à diverses généralisations de la méthode de R.DAVID.
Notre première contribution a été l'écriture de programmes suivant la méthode initiale de DAVID. Ces programmes ont montré la grande supériorité de la méthode sur toutes les précédentes tant au point de vue complexité du programme que du point de vue du coût statistique des réseaux obtenus. Toutefois, certaines faiblesses de la méthode nous sont apparues à l'écriture des programmes.
La première est de nécessiter au départ une représentation sous forme de tableau d'états primitifs, ce qui peut parfois être impossible pour les gros systèmes. Nous proposons de pallier à cette difficulté en utilisant une méthode de consensus pour grouper les états, tout en obtenant une représentation minimale même si celle de départ n'était pas primitive.

Cette méthode n'a pas encore été exploitée sur ordinateur, mais sa programmation étant analogue à celle du consensus en algèbre de Boole, ne posera pas de difficultés.

Une deuxième faiblesse de la méthode concerne la recherche des relations d'hypovalence. Celles-ci peuvent en effet ne pas exister si les sorties sont quelconques. Nous proposons en fait une méthode de codage des sorties, dérivée de la méthode de synthèse topologique : le principe est de considérer une sortie comme une fonction booléenne des entrées et des variables internes, et de fractionner cette fonction en s'efforçant d'utiliser le plus possible les variables d'entrées ou certaines variables internes "peu coûteuses". Comme précédemment, cette méthode n'a pas encore été programmée.

Enfin, nous présentons une méthode de décomposition des gros systèmes séquentiels en sous systèmes ayant peu d'interconnexions entre eux, et ceci dans des cas où la méthode d'Hartmanis ne donne aucuns résultats.

CHAPITRE 2

Outils mathématiques de base

THEORIE DES FONCTIONS INCOMPLETES

Introduction.

En pratique, les fonctions logiques rencontrées dans les systèmes sont en général incomplètement spécifiées. Il s'ensuit que certaines théories classiques de l'algèbre de Boole sont inapplicables. En particulier, une propriété fondamentale disparait : l'ensemble des fonctions logiques incomplètes ne forment pas une algèbre de Boole. Il a donc été nécessaire d'approfondir la théorie de ces fonctions.

2 - 1 Etude générale des fonctions incomplètes.

2-1-1 Définition et notations.

Une fonction logique incomplète peut être caractérisée par la donnée de deux fonctions booléennes dénommées bornes inférieures et supérieures :

$$\text{On notera } F = \{\underline{f},\ \overline{f}\} \quad \text{avec } \underline{f} \leq \overline{f}$$

Pour certaines démonstrations, nous utiliserons la notation :

$$F = \underline{f} + \emptyset\ \overline{f}$$

où le symbole $\emptyset$ désigne une quantité indéterminée 0 ou 1.

Dans les problèmes pratiques, il est souvent plus commode de considérer non pas la borne supérieure $\overline{f}$ mais son complément $\overline{f}'$. On utilisera alors une autre notation qui sera appelée "Normale" :

$$F = [\underline{f},\ \overline{f}']$$

notation qui fait mieux ressortir la symétrie entre les grandeurs 0 et 1, et qui se distingue de la précédente par la présence de crochets au lieu d'accolades.

Notations : Dans tout ce qui suit, on notera en caractères minuscules les fonctions booléennes complètes, et en majuscules les fonctions incomplètes, exception faite pour l'ensemble des variables $x_1\ x_2\ \ldots x_n$ qui sera noté X.

Remarque : Une fonction complète f pourra en général être assimilée à la fonction incomplète particulière F = [f, f'] . Toutefois, il conviendra de prendre certaines précautions, les opérations sur les fonctions incomplètes n'ayant pas toutes les propriétés des opérations de l'algèbre de Boole.

2-1-2 Opérations.

On définit des opérations qui sont des extensions des opérations booléennes. Ces opérations seront notées de la même manière tant qu'il n'y a pas de confusion possible. (Les grandeurs complètement et incomplètement spécifiées n'étant pas, elles, notées de la même manière).

- COMPLEMENT -

On définit le complément F' de la fonction $F = [\underline{f}, \overline{f}']$ par $F' = [\overline{f}', \underline{f}]$. Cette opération est une extension de la notion habituelle de complément ; si F est complète, on a en effet :

$$F = [f, f'] \qquad \text{et} \qquad F' = [f', f]$$

F' est bien une fonction incomplète puisque si $\underline{f} \leq \overline{f}$ on a bien $\overline{f}' \leq \underline{f}'$

Remarque : C'est la simplicité de la définition dans la notation normale qui nous la fait préférer aux autres notations.
Si on note par exemple :

$$F = \{\underline{f}, \overline{f}\} \quad \text{alors} \quad F' = \{(\overline{f})' , (\underline{f})'\}$$

et le calcul de F' nécessite deux calculs de compléments. En particulier, la représentation normale sera exclusivement utilisée pour la représentation sur ordinateur des fonctions incomplètes.

- UNION -

Par définition, l'union de deux fonctions $F = [\underline{f}, \overline{f}']$ et $G = [\underline{g}, \overline{g}']$ est la fonction :

$$F + G = [\underline{f} + \underline{g} , \overline{f}' . \overline{g}']$$

opération qui est, comme ci-dessus, une extension de l'union habituelle puisque si $F = [f, f']$ et $G = [g, g']$ on a bien $H = [f+g, f' . g']$

Cette fonction est bien une fonction incomplète puisque

$$\text{si } \underline{f} \leq \overline{f} \quad \text{et} \quad \underline{g} \leq \overline{g} \quad \text{on a} \quad \underline{f} + \overline{g} \leq \overline{f} + \overline{g}$$

- INTERSECTION -

De la même manière on définit l'intersection de F et G par

$$F . G = [\underline{f} . \underline{g} , \overline{f}' + \overline{g}']$$

comme précédemment cette fonction F G est bien une fonction incomplète.

Propriétés de ces opérations.

On montrerait par un calcul élémentaire que les opérations union et intersection sont associatives commutatives, idempotentes, distributives l'une par rapport à l'autre.

On peut grouper ces diverses propriétés en énonçant le théorème suivant.

Théorème 1.

Les fonctions incomplètement forment un treillis distributif pour les opérations d'union et d'intersection.

L'élément maximum du treillis est [1, 0]

L'élément minimum du treillis est [0, 1]

Remarque :

On peut démontrer que ce treillis n'est pas un treillis de Boole car il n'est pas complémenté :

à tout F = [a,b] on ne peut en effet pas forcément associer F" = [x,y] tel que $F + F'' = [1,0]$ et $F \cdot F'' = [0,1]$

On aurait en effet alors :

$a + x = 1 \qquad b.y = 0$

$a.x = 0 \qquad b + y = 1$

d'où on tire

$x = a' \quad , y = b'$

mais si F est strictement incomplète, on a $a < b$ et donc $x > y$

et le couple [x,y] ne représente pas une fonction incomplète.

DISJONCTION

La disjonction de F et G sera par définition :

$$F \oplus G = F \cdot G' + F' \cdot G$$

ce qui est bien une fonction incomplète puisque complément produit et somme donnent des fonctions incomplètes.

Ceci peut se développer en appliquant les règles précédentes :

$$F \oplus G = [\underline{f}, \overline{f}'] \cdot [\overline{g}', \underline{g}] + [\overline{f}', \underline{f}] \cdot [\underline{g}, \overline{g}']$$
$$= [\underline{f}\overline{g}' + \overline{f}'\underline{g} \ , \ (\overline{f}' + \underline{g}) \cdot (\underline{f} + g')]$$

opération qui comme précédemment, est une extension de la disjonction en algèbre de Boole.

On vérifie de manière très simple les propriétés suivantes :

- la disjonction est associative et commutative :

$$(F \oplus G) \oplus H = F \oplus (G \oplus H) \text{ et } F \oplus G = G \oplus F$$

- elle possède un élément neutre [0, 1]. En effet :

$$[a, b] \oplus [0,1] = [a \cdot 1 + b \cdot 0, \ a \cdot 0 + b \cdot 1] = [a,b]$$

- par ailleurs $F \oplus [1, 0] = F'$

En effet $[a, b] \oplus [1, 0] = [a \cdot 0 + b \cdot 1, \ a \cdot 1 + b \cdot 0]$

$= [b, a] = [a, b]'$

Mais certaines propriétés ne sont pas vérifiées.

On n'a pas en général :

$$(F \oplus G)H = F.H \oplus GH$$

Il n'y a pas distributivité de l'intersection pour la disjonction.

En effet, si $F = \left[a, b\right]$, $G = \left[c, d\right]$ et $H = \left[e, f\right]$

On a :

$$(F \oplus G)H = \left[(ad + bc)e,\ ac + bd + f\right]$$

$$\text{et} \quad FH + GH = \left[(ad + bc)e,\ ace + bd + f\right]$$

Et, en général, les deux bornes supérieures ne sont pas égales.

Par ailleurs, il n'existe pas d'inverse pour la disjonction.

On n'a pas $F \oplus F = \left[0, 1\right]$

En effet $\left[a, b\right] \oplus \left[a, b\right] = \left[ab + ba,\ aa + bb\right]$

$$= \left[0,\ a + b\right]$$

Et si F est une fonction strictement incomplète, $a \neq b'$ et donc $a + b \neq 1$

On peut se poser la question : Existe-t'il un élément F" inverse de F ?

En fait, il n'en est rien si F n'est pas complet. En effet, supposons qu'il existe $\left[x, y\right]$ inverse de $\left[a, b\right]$. On aurait :

$$\left[a, b\right] + \left[x, y\right] = \left[0, 1\right]$$

donc $\left[ay + bx,\ ax + by\right] = \left[0, 1\right]$

donc $ay + bx = 0$

$ax + by = 1$

système d'équations qui se réduit à l'équation unique :

$$ay + bx + a'b' + a'y' + x'b' + x'y' = 0$$

$$x(y + b + a') + x'(y' + b' + a) = 0$$

La théorie des équations booléennes (cf [1]) nous montre que cette équation n'est possible que si les coefficients de x et de x' sont disjoints :

$$(y + b + a')(y' + b' + a) = 0$$

C'est-à-dire :

$$y(b + a') + y'(b' + a) = 0$$

équation qui n'est encore possible que si

$$(b + a')\ (b' + a) = 0$$

$$b \oplus a' = 0$$

$$b = a'$$

Pour qu'une fonction ait un inverse pour la disjonction, il faut et il suffit qu'elle soit complète.

2-1-3 Spécification.

Définitions.

1) On dira qu'une fonction $G = \left[\underline{g}, \overline{g}'\right]$ est plus spécifiée que $F = \left[\underline{f}, \overline{f}'\right]$ si on a :

$$\underline{f} \leq \underline{g} \quad \text{et} \quad \overline{f}' \leq \overline{g}'$$

On notera cette relation :

$$\boxed{F >> G}$$

Cette relation est bien une relation d'ordre car elle est le produit de deux relations d'ordre.

On dira également que F est plus incomplète que G (l'ensemble des valeurs des variables pour lesquelles F est défini est contenu dans l'ensemble où G est défini)

2) Si G est fonction complète $G = \left[g, g'\right]$, on dira que g est une solution de F.

Chercher à réaliser physiquement une fonction incomplète F, reviendra donc à chercher un réseau logique réalisant une solution de F.

2-1-4 Expression des fonctions incomplètes, forme de Lagrange.

Toute fonction F(x) incomplète peut se mettre sous la forme :

$$\boxed{F(x) = x \,.\, F(1) + x' \,.\, F(0)}$$

en confondant ici la variable x et la fonction incomplète $\left[x, x'\right]$. En effet

$$xF(1) + x'F(0) = \left[\underline{f}(1), \overline{f}'(1)\right] \,.\, \left[x, x'\right] + \left[\underline{f}(0), \overline{f}'(0)\right] \,.\, \left[x', x\right]$$

$$= \left[xf(1), x' + \overline{f}'(1)\right] + \left[x'\underline{f}(0), x + \overline{f}'(0)\right]$$

$$= \left[x\underline{f}(1) + x'\underline{f}(0), \; (x' + \overline{f}'(1))(x + \overline{f}'(0))\right]$$

$$= \left[x\underline{f}(1) + x'\underline{f}(0), \; x\overline{f}'(1) + x'\overline{f}'(0)\right]$$

$$= \left[\underline{f}(x), \overline{f}'(x)\right]$$

2-1-5 Enveloppes

Définition.

On appelle "enveloppe" ou "fermeture" une transformation $f \to \varphi(f)$ extensive, monotone, idem-potente. (Bibliographie n°[3]).

Dans le cas des fonctions incomplètes, on appellera donc enveloppe une application φ de l'ensemble des fonctions incomplètes dans lui-même, telle que :

a) $\forall F$, $\varphi(F) << F$: la transformation est extensive

b) si $F << G$, $\varphi(F) << \varphi(G)$: l'application est monotone

(la relation << étant celle définie au paragraphe 2-1-3

c) $\varphi(\varphi(F)) = \varphi(F)$ l'application est idem-potente

Exemple : la transformation qui, à F, associe, si elle existe, la plus grande fonction croissante compatible avec F, est une enveloppe.

Enveloppe intérieure ou "réduite".

Etant donnée $F = \left[\underline{f}(X), \overline{f}'(Y)\right]$

posons $Z = X \cap Y$

On définit la réduite r(F) par :

$$r(F) = \left[\underline{f}(X)^{Z}, \overline{f}'(Y)^{Z}\right]$$

en utilisant la notation des enveloppes indépendantes de certaines variables utilisée par Lapscher (cf n°[3]), cette transformation r est bien une enveloppe puisque :

a) r(f) est par définition compatible avec f donc $r(f) \leq f$

b) $r(r(F)) = \left[\underline{f}^{Z}{}^{Z}, \overline{f}'^{Z}{}^{Z}\right]$

et la propriété d'enveloppe est vraie pour la transformation $f \to r(f)$

Remarque.

1) Cette enveloppe offre un intérêt pratique évident : la réduite d'une fonction F, dépendant de moins de variables que F, sera plus simple que F.

2) On démontre (cf. thèse de l'auteur : bibliographie n°[14]) que la réduite r(F) d'une fonction F existe toujours bien qu'on ait parfois r(F) = F

2-1-6 Distances.

Par extension de la notion de distance habituellement utilisée en algèbre de Boole, on appellera distance entre $F = [a, b]$ et $G = [c, d]$ la quantité :

$$d(F, G) = |ad + bc|$$

C'est le nombre de valeurs des variables pour lesquelles $F \neq G$

Cette quantité vérifie :

- d(F, G) est positif
- d(F, G) = 0 implique qu'il n'y a pas de valeur des variables pour lesquelles F est sûrement différent de G, donc il existe une fonction H compatible avec F et G.

Cette grandeur n'est pas une distance au sens mathématique car l'inégalité triangulaire n'est pas vérifiée.

distance d'une fonction incomplète à une fonction complète :

si $F = [\underline{f}, \overline{f}']$ et $G = [g, g']$

On a, en identifiant G à g :

$$d(F, g) = |\underline{f}g' + \overline{f}'g|$$

et d(F, g) = 0 implique que g est solution de F:

En effet $\underline{f}.\, g' + \overline{f}'.g = 0$ implique

$\underline{f}.g' = 0$ et $\overline{f}'.g = 0$ donc

$\underline{f} \leq g$ et $g \leq \overline{f}$

2-1-7 Variations des fonctions

2-1-7-1 Définitions.

Etant donnée $f(x_i, X)$ une fonction booléenne, on définit les fonctions suivantes :

dérivée : on pose

$$\lambda_i(f(X)) = f(0, X) \oplus f(1, X)$$

c'est la "dérivée" en x_i de f(x, X)

Soit f(xyz) = xy + yz + zx

On a alors :

$$\begin{aligned} \lambda_x(f(xyz)) &= f(0, yz) \oplus f(1, yz) \\ &= (yz) \oplus (y + yz + z) \\ &= y \oplus z \end{aligned}$$

et de la même manière puisque f est symétrique en x y et z :

$$\lambda_y(f(xyz)) = x \oplus z$$

$$\lambda_z(f(xyz)) = x \oplus y$$

enveloppe inférieure

On pose : $\boxed{\mu_i(f) = f(0, X) \,.\, f(1, X)}$

c'est l'enveloppe inférieure indépendante en x_i de f (plus grande fonction indépendante de x_i et inférieure à f). En effet

a) le produit f(0).f(1) est inférieur à $f(x_i)$, et indépendant de x_i

b) si on suppose g(X) indépendant de x_i et inférieur à $f(x_i, X)$, on a :

pour x = 0 , $g(X) \leq f(0, X)$

pour x = 1 , $g(X) \leq f(1, X)$

donc par produit $g(X) \leq f(0, X).f(1, X)$

donc $g(X) \leq \mu_1(f)$ ce qui est la propriété cherchée.

Exemple : f = xy + yz + zx

On a : $\mu_x(f(xyz)) = (yz) \,.\, (y + yz + z)$

$= y.z$

et, de même, par raison de symétrie :

$$\mu_y(f(xyz)) = x.z$$

$$\mu_z(f(xyz)) = x.y$$

enveloppe supérieure

On pose : $\boxed{\theta_i(f(X)) = f(0, X) + f(1, X)}$

On démontre comme précédemment que $\theta_i(f)$ est l'enveloppe supérieure indépendante en x_i de f.

Exemple : f = xy + yz + zx

$$\theta_x(f(xyz)) = (yz) + (x + yz + z)$$

$$= y + z$$

$$\theta_y(f(xyz)) = x + z$$

$$\theta_z(f(xyz)) = x + y$$

frontière de f(X)

Etant donnée un sous ensemble X1 de X, on appelle frontière en X1 de f(X) la fonction

$$\underset{X1}{\text{front}}\,(f) = f(X) \cdot \sum_{x_i \in X_1} \lambda_i(f)$$

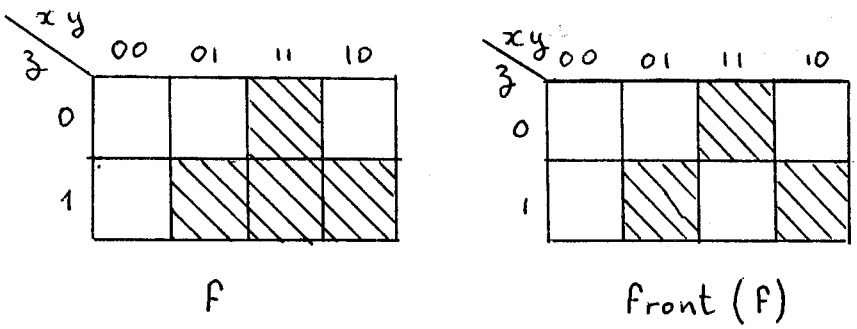
Nous dirons que la frontière est l'ensemble des points de f ayant un voisin en x_i dans f'.

Si $X_1 = X$ on dira que la frontière est totale.

Si $X_1 = \{x_i\}$ la frontière se réduit à $\lambda_i(f(X)) . f(X)$

Exemple : $f(xyz) = xy + yz + zx$

$$\underset{X}{\text{front}}\,(f) = (xy + yz + zx)((y \oplus z) + (x \oplus z) + (x \oplus y))$$
$$= (xy + yz + zx)(yz' + y'z + xz' + zx' + xy' + yx')$$
$$= xyz' + xy'z + x'yz$$



Voisinage de f(X)

On appellera ainsi la frontière de f'(X). On aura donc :

$$\underset{X1}{\text{Voisin}}\,(f(X)) = f'(X) \cdot \sum_{x_i \in X1} \lambda_i(f(X))$$

Variation de f(X)

On appellera ainsi l'ensemble des points frontière de f et de f' :

$$\underset{X1}{\text{Var}}(f(X)) = \sum_{x_i \in X1} \lambda_i(f(X))$$

La variation vérifie manifestement les propriétés suivantes :

$$\underset{X_1}{Var}(f) = \underset{X_1}{Var}(f')$$

$$\underset{X1}{Var}(f) = \underset{X1}{Voisin}(f) + \underset{X1}{front}(f)$$

Intérieur de f(X)

On appellera intérieur de f(X) l'ensemble des points de f(X) qui ne sont pas frontière en X_1 :

$$\boxed{\underset{X_1}{Int}(f(X)) = f \,.\, (front(f(X)))'}$$

C'est également l'ensemble des points dont tous les voisins sont dans f. On a donc :

$$\boxed{\underset{X_1}{Int}(f(X)) = \prod_{x_i \in X_1} \mu_i(f(X))}$$

Cette formule est plus intéressante que la précédente car dans certains cas elle donne lieu à un calcul aisé:

Si f(X) est donnée sous forme de base première complète, et si x_i est une variable de X_1, on écrira :

$$f(X) = A_i x_i + B_i x_i' + C_i$$

$$\mu_i(f) = f(0).f(1) = (B_i + C_i)(A_i + C_i)$$

$$= C_i + A_i B_i$$

Mais $A_i B_i$ est formé de monômes qui sont consensus d'un monôme de $A_i x_i$ et d'un monôme de $B_i x_i'$; ce consensus est donc dans C_i et on a :

$$\underline{\mu_i(f) = C_i}$$

et donc

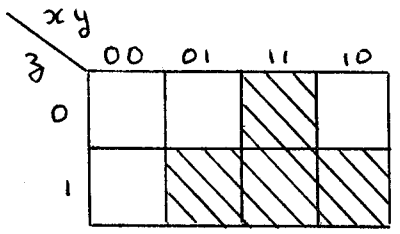
$$\underset{X_1}{Int}(f(X)) = \prod_{x_i \in X_1} C_i$$

Exemple

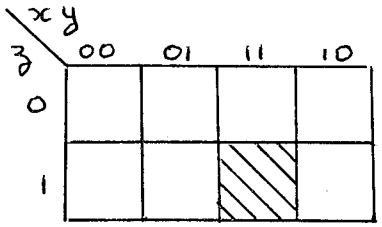
$$f(xyz) = xy + yz + zx$$

$$int(f) = (yz).(xz).(zx)$$

$$= xyz$$



f



int (f)

Adhérence de f(X).

On appellera adhérence la fonction :

$$\underset{X_1}{\text{adher}}(f(X)) = f + \underset{X_1}{\text{Voisinage}}(f)$$

Son complément est l'intérieur de f' et on a :

$$\text{adher}\ (f(X)) = \left(\prod_i \mu_i(f')\right)'$$

$$= \sum (\mu_i(f'))'$$

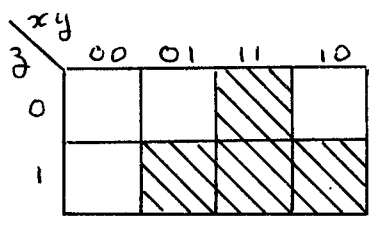
$$\sum \Theta_i(f)$$

$$\underset{X_1}{\text{adher}}(f(X)) = \sum_{x_i \in X_1} \Theta_i(f(X))$$

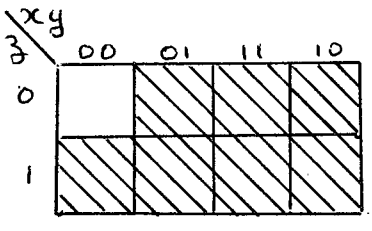
Exemple : $f(X) = xy + yz + zx$

$$\text{adher}(f) = (x + y) + (y + z) + (Z + x)$$

$$= x + y + z$$



f



Adher (f)

2-1-7-2 Propriétés de l'adhérence et de l'Intérieur.

Lemme 1

$$\boxed{\underset{X_1}{\text{adher}}(f(X)) + \underset{X_1}{\text{adher}}(g(X)) = \underset{X_1}{\text{adher}}(f(X) + g(X))}$$

En effet :

$$A \in \text{adher}(f+g) \Longleftrightarrow \exists B,\ d(A, B) \leq 1 \text{ et } f(B) + g(B) = 1$$

$$\Longleftrightarrow \exists B,\ d(a, b) \leq 1 \text{ et } f(B) = 1$$

ou

$$\exists B,\ d(a, b) \leq 1 \text{ et } g(B) = 1$$

$$\Longleftrightarrow A \in \text{adher}(f) + \text{adher}(g)$$

Lemme 2

$$\boxed{f(X) \leq g(X) \Leftrightarrow \underset{X_1}{\text{adher}}(f(X)) \leq \underset{X_1}{\text{adher}}(g(X))}$$

En effet : $f(X) \leq g(X) \quad h(X),\ f(X) + h(X) = g(X)$

Et donc d'après le lemme précédent :

$$\text{adher}(f(X)) + \text{adher}(h(X)) = \text{adher}(g(X))$$

Et enfin $\text{adher}(f(X)) \leq \text{adher}(g(X))$

Lemme 3

$$\boxed{\text{adher}(f(X).g(X)) \leq \text{adher}(f).\ \text{adher}(g)}$$

En effet $f.g \leq f \Longrightarrow \text{adher}(f.g) \leq \text{adher}(f)$

$$f.g \leq g \Longrightarrow \text{adher}(f.g) \leq \text{adher}(g)$$

et par produit $\text{adher}(f.g) \leq \text{adher}(f).\text{adher}(g)$

Par ailleurs, il n'y a pas toujours égalité comme le montre l'exemple :

$$\text{adher}(f)\ .\ \text{adher}(f') = \text{Var}(f)$$

$$\text{adher}(f.f') = \text{adher}(0) = 0$$

Adhérence d'une fonction incomplète

Si $F(X) = \{\underline{f}(X), \overline{f}(X)\}$.

On appellera adhérence de F(X) la fonction incomplète

$$\text{adher}(F(X)) = \{\text{adher}(\underline{f}(X)), \text{adher}(\overline{f}(X))\}$$

(c'est bien une fonction réelle puisque $\underline{f} \leq \overline{f}$ et donc $\text{adher}(\underline{f}) \leq \text{adher}(\overline{f})$)

Théorème.

L'adhérence d'une fonction g, compatible avec F, est compatible avec adher(F). En effet, d'après le lemme 2,

$$\underline{f} \leq g \leq \overline{f} \Longrightarrow \text{adher}(\underline{f}) \leq \text{adher}(g) \leq \text{adher}(\overline{f})$$

Lemme 4

$$f(X) \leq g(X) \Longrightarrow \underset{X_1}{\text{Int}}(f(X)) \leq \underset{X_1}{\text{Int}}(g(X))$$

Ce lemme se déduit du lemme 2:

Si $f(X) \leq g(X)$ on a $f'(X) \geq g'(X)$

et donc $\underset{X_1}{\text{Adher}}(f'(X)) \geq \underset{X_1}{\text{Adher}}(g'(X))$

Et, par complémentation :

$$(\underset{X_1}{\text{Adher}}(f'(X)))' \leq (\underset{X_1}{\text{Adher}}(g'(X)))'$$

Ou encore $\text{Int}(f(X)) \leq \text{Int}(g(X))$

Corollaire.

Comme précédemment on en déduit

1) que l'intérieur d'une fonction incomplète $F = \{\underline{f}, \overline{f}\}$ peut être défini comme

$$\underset{X_1}{\text{Int}}(F) = \{\underset{X_1}{\text{Int}}(\underline{f}), \underset{X_1}{\text{Int}}(\overline{f})\}$$

2) que si g est compatible avec F alors $\underset{X_1}{\text{Int}}(g)$ est compatible avec $\underset{X_1}{\text{Int}}(F)$

Applications.

Ces notions seront utiles dans la détermination de réseaux sans aléas et de réseaux testables (chapitres 3-4 et 3-5).

C H A P I T R E 3

CONCEPTION DES RESEAUX COMBINATOIRES

Introduction : les méthodes peuvent être classées en trois types.

a - Méthodes par factorisation: [11], [17], [18]

Etant donnée une fonction f par une forme polynomiale, on déduit de cette forme un réseau d'opérateurs NOR (ou NAND ou ET/OU) par mises en facteurs et complémentations successives. Ces méthodes dérivées des réalisations en somme de produits sont simples mais ne conduisent pas à des réalisations économiques : les réseaux formés sont arborescents : si plusieurs fonctions doivent être réalisées, il est difficile de déterminer une partie commune dans leurs réseaux respectifs.
Cette méthode est plus destinée à être utilisée "à la main" que sur ordinateur, nous ne l'avons pour ainsi dire, pas étudiée.

b - Méthodes par composition ou méthodes ascendantes.

Ces méthodes consistent à former tous les réseaux non redondants par coûts croissants jusqu'à obtention d'un réseau réalisant la fonction donnée. Cette méthode, très séduisante parce qu'elle conduit à des réseaux minimaux, n'est envisageable que pour des réalisations relativement simples. Elle serait inapplicable pour des complexités dépassant une vingtaine de portes. (Bibliographie : [1], [13] et [21])

c - Méthodes par "fractionnement" ou "descendantes". [14]

Au lieu de composer des fonctions simples (les variables) jusqu'à obtention des fonctions à réaliser, on peut concevoir une méthode procédant en sens inverse : la fonction donnée sera décomposée en sous fonctions plus simples, jusqu'à obtention des variables.

Définition : "fractionner" une fonction à réaliser f(x) à l'aide d'un opérateur A, à k entrées revient à déterminer les fonctions intermédiaires plus simples que f et notées $f_1 \ldots\ldots f_k$, telles que :

$$f = A(f_1\ f_2 \ldots\ldots\ldots\ldots f_k)$$

et à itérer le procédé pour chacun des f_i jusqu'à l'obtention de fonctions disponibles, variables ou fonctions déjà réalisées dans une étape précédente. Cette méthode de calcul qui nous est propre, a déjà reçu diverses

interprétations : dans une thèse antérieure nous avons développé deux méthodes de fractionnement déduites de la théorie des formes polynomiales des fonctions booléennes.

Dans ce qui suit nous rappellerons brièvement une de ces méthodes, en en montrant quelques aspects intéressant dans le domaine des aléas et des tests. Ensuite nous donnerons une méthode de fractionnement nouvelle, spécialement étudiée en vue d'une intégration à grande et moyenne échelle.

3 - 1 Méthode de fractionnement littérale.

La méthode que nous exposons ici a déjà donné lieu à une thèse de docteur-ingénieur soutenue par l'auteur de celle-ci en 1967 (Bibliographie [14]). Nous en rappellerons cependant les principes.

3-1-1 Opérateurs.

Les méthodes de fractionnement littérales sont très générales et s'appliquent à des opérateurs très variés, mais ayant la propriété d'être privilégiés.

Définition. On appelle fonction booléenne privilégiée, une fonction $f(x_1, x_2 \ldots\ldots x_n)$ ayant une expression à l'aide des signes union, intersection et complémentation dans laquelle chaque variable n'apparait qu'une fois.
Un opérateur sera dit privilégié s'il réalise une telle fonction.

Exemples. Les opérateurs NAND et NOR sont privilégiés, de même que les opérateurs réalisant les fonctions $(x + y'z')w'$, $x'y' + w'z'$ etc.....
Par contre, ne sont pas privilégiés, les opérateurs majorité et disjonction qui s'écrivent $(x + y)z + xy$ et $xy' + x'y$.

Propriété. Tous les opérateurs privilégiés se prêtent bien aux méthodes de fractionnement littérales. En fait, nous exposerons la méthode de fractionnement sur l'opérateur NOR qui est le plus utilisé. Nous donnerons à la fin de ce chapitre quelques exemples de généralisation .

3-1-2 "Base" de fractionnement.

Par extension de la terminologie des espaces vectoriels, nous appellerons "base" un ensemble de variables ou fonctions pouvant générer par composition des opérateurs donnés, la ou les fonctions à réaliser. Une telle base peut être minimale : l'ensemble des variables dont dépend effectivement la fonction $F(x_1 \ldots . x_n)$ constitue une telle base.
En réalité, une telle notion n'est pas intéressante pour la conception des réseaux. Il est souvent plus intéressant de minimiser le nombre d'opérateurs nécessaires que de minimiser le nombre de générateurs de la base. Si, par exemple, une première fonction F utilise pour sa réalisation des fonctions intermédiaires F_i, il peut être intéressant d'adjoindre ces fonctions à la base constituée des variables, pour le fractionnement d'une deuxième fonction G.
Cette notion de "base" est indépendante de la méthode de fractionnement choisie, elle interviendra en particulier dans la méthode dite "topologique".

3-1-3 Principe du fractionnement.

Etant donnée une fonction incomplète

$$F(x_1 \ldots\ldots x_n) = \left[\underline{f} \ , \ \overline{f'}\right]$$

et des opérateurs NOR à k entrées, nous recherchons k fonctions intermédiaires notées $F_1, F_2, \ldots\ldots\ldots F_k$ telles que

$$F = \text{NOR}\ (F_1, F_2, \ldots\ldots\ldots\ldots F_k)$$

ou encore

$$F' = \sum_{i=1}^{k} F_i \qquad (1)$$

équation qui se décompose en deux :

$$(\overline{f})' = \sum_{i=1}^{k} \underline{f_i} \quad (2) \qquad (\underline{f})' = \sum_{i=1}^{k} \overline{f_i} \quad (3)$$

Il est évident que chaque F_i sera d'autant plus simple à réaliser qu'il est plus incomplet : chaque $\underline{f_i}$ sera donc choisi aussi petit que possible, et chaque $\overline{f_i}$ aussi grand que possible.

Détermination des $\underline{f_i}$

Supposons (f)' donné sous la forme polynomiale

$$\overline{f}' = \sum_{j \in J} m_j$$

Soit $J_1, J_2, \ldots\ldots\ldots\ldots J_k$ un recouvrement de l'ensemble d'indice J :

$$J_1 \cup J_2 \cup \ldots\ldots \cup J_k = J$$

On définira $\underline{f_i}$ par :

$$\underline{f_i} = \sum_{j \in J_i} m_j \qquad (4)$$

On aura bien alors :

$$\sum_{i=1}^{k} \underline{f_i} = \sum_{j \in J_1} m_j + \ldots\ldots + \sum_{j \in J_k} m_j = \sum_{j \in J} m_j$$

Et l'égalité (2) est bien vérifiée.

Détermination des $\overline{f_i}$.

On choisira la plus grande valeur possible pour chaque $\overline{f_i}$, c'est-à-dire

$f_i = (\underline{f})'$ pour tout i, et on aura bien

$$\sum_{i=1}^{k} \overline{f_i} = \sum_{i=1}^{k} (\underline{f})' = (\underline{f})' \text{ et l'équation (3) est vérifiée}$$

3-1-4 Convergence.

La méthode converge si chaque F_i est plus "simple à réaliser" que F : il faut pouvoir affirmer qu'après un certain nombre d'itérations du fractionnement, les fonctions obtenues seront réalisables par des éléments de la base. Deux critères principaux sont utilisés pour assurer la convergence : un critère basé sur le niveau de spécification des fonctions, et un critère basé

sur le nombre de variables des fonctions.

3-1-4-1 Niveau de spécification.

On appellera "niveau de spécification" ou plus simplement "niveau" d'une fonction incomplète, le nombre de valeurs des variables $x_1 \ldots\ldots x_n$, où la valeur de $F(x_1 \ldots\ldots x_n)$ est définie.

Théorème.

Si à chaque fractionnement le niveau de chaque F_i est strictement inférieur au niveau de F, alors la méthode converge.

En effet, après un certain nombre de fractionnement, on arrivera à des fonctions F_i élémentaires ayant un seul monôme canonique dans sa borne inférieure, et un seul dans sa borne supérieure. A ce stade, le fractionnement devient impossible, mais toute variable x_j présente sous une forme dans $\underline{f_i}$ et sous l'autre dans $\overline{f_i}$ est une solution de F_i:

Si $\underline{f_i} = x.m_i$ et $\overline{f_i}' = x'.n_i$

alors on a $x.m_i \leq x \leq x + (n_i)'$

Par ailleurs, une telle variable existe toujours à ce stade. Si en effet il existait F_i dont les deux bornes vérifient

$$\underline{f_i} = m_i$$

$$(\overline{f_j})' = n_i$$

m_i et n_i étant deux monômes tels qu'il n'existe aucune variable accentuée dans l'un et non dans l'autre, le produit $m_i.n_i$ ne serait pas nul ce qui est exclu puisque par définition $\underline{f_i}$ et $(\overline{f_i})'$ sont disjoints.

Remarque. Ce théorème justifie a posteriori la convention de fractionnement qui imposait à chaque F_i d'être le plus incomplet possible.

Corollaire 1. On imposera, lors du choix de la partition des monômes de $(f)'$ que cette fonction soit écrite sous forme irredondante. En effet, supposons $(\overline{f})'$ écrit sous forme redondante :

$$(\overline{f})' = \sum m_i + \sum m_j$$

$\Sigma\ m_i$ constituant une base irredondante et $\Sigma\ m_j$ des monômes superflus.
Il existe alors un fractionnement particulier de $(\overline{f})'$ en deux fonctions f_1 et f_2:

$$f_1 = \Sigma\ m_i = (\overline{f})$$

$$f_2 = \Sigma\ m_j$$

et alors $F_1 = F'$ et cette fonction n'est pas de niveau strictement inférieur à F.

Corollaire 2. En vue d'avoir une rapide convergence on choisira dans le découpage de la base de $(\overline{f})'$ une partition des monômes et non un recouvrement quelconque, en imposant, de plus, d'avoir une partition en au moins 2 ensembles.

Définition : On appellera "forme" une grandeur qui est soit une variable soit son complément sans que l'alternative soit précisée.

3-1-4-2 Nombre de variables

Théorème : Si la base du fractionnement comporte les variables $x_1 \ldots x_n$ et si $F(x_1 \ldots x_n)$ est fractionné en sous fonctions F_i s'écrivant chacune avec moins de formes que F alors la méthode converge.
En effet, après au plus 2×n-fractionnements, les fonctions intermédiaires dépendront d'une seule variable et elles seront résolues par cette variable, ou son complément.

Applications

3-1-4-2-1 Méthode de CHEIN

On peut rechercher les décompositions en sommes de la fonction $(\overline{f})'$. Cette méthode a été appliquée par CHEIN (cf n° 15) en utilisant des méthodes de graphes.

3-1-4-2-2 Méthode des réduites

On peut remplacer en pratique une fonction intermédiaire par sa réduite.

En effet, il est très rare qu'une fonction ayant une solution à n variables, en ait une moins coûteuse en utilisant un plus grand nombre de variables. On peut alors définir un fractionnement de la manière suivante :

a) F est fractionné suivant les méthodes précédentes en F_1 F_2..... F_k

b) Chacune de ces fonctions est remplacée par sa réduite.

c) On itère le procédé de fractionnement sur les réduites $r(F_i)$.

Diverses méthodes permettant d'assurer que la réduite $r(F_i)$ n'est pas égale à F_i, sont envisageables. Citons une méthode élémentaire qui a l'avantage d'être extrêmement rapide : désignons par $\tilde{x}$ une occurence accentuée ou non d'une variable x. Une telle quantité sera appellée une "forme".

Définition : on appelle fréquence d'une forme $\tilde{x}$ dans une expression particulière de F le produit N = N1 . N2, N1 étant le nombre d'occurences de $\tilde{x}$ dans la base choisie pour f., N2 étant le nombre d'occurences de la forme complémentaire $(\tilde{x})'$ dans la base choisie pour $(\bar{f})'$.

Exemple : soit F = xy + xz + x'z', x'z + xy'z'
Dans cette expression de F la fréquence de x sous forme directe est
N1 = 2 : x apparait 2 fois dans f,
N2 = 1 : x' apparait 1 fois dans $(\bar{f})'$, donc $N_x = 2$

Fractionnement suivant la variable la plus fréquente.
Soit $\tilde{x}$ la forme la plus fréquente dans F (Si plusieurs variables présentent la même fréquence maximale, on prendra la première par ordre alphabétique, par exemple).
Définissons le fractionnement suivant :
- La fonction F_1 aura comme borne inférieure la somme des monômes de $(\bar{f})'$ contenant $(\tilde{x})'$

- Les fonctions $F_i (i \geq 2)$ formeront une partition des monômes restant.

Propriétés de ce fractionnement :

- Les fonctions F_i, $(i \geq 2)$ dépendent d'une forme de moins que F, donc la condition de convergence sera vérifiée pour leurs réduites.
- La fonction F_1 peut éventuellement dépendre du même nombre de variables que F mais au cours de l'itération de la méthode pour F_1, x sera la forme la plus fréquente puisque son nombre d'occurence n'a changé ni dans $(\overline{f_1})'$ qui est par définition $\underline{f}$ ni par construction dans f_1. Ainsi, on génèrera à ce stade k-1 fonctions $F_{12}\ F_{13}\ldots F_{1(k-1)}$ ne dépendant plus de $\tilde{x}'$ et une fonction F_{11} ayant comme solution $\tilde{x}$ en facteur dans sa borne inférieure, et $(\tilde{x})'$ en facteur dans sa borne supérieure.

On a ainsi, au plus par deux itérations, imposé les conditions de convergence.

3-1-5 Augmentation de la base.

A chaque étape de la méthode, le but cherché est de trouver une solution f de la fonction F à réaliser (qui peut être une fonction intermédiaire ou une des fonctions données). F peut en fait être résolue pour deux raisons.

- On peut avoir trouvé dans la base une solution de F.
- On peut avoir fractionné F en K fonctions F_i qui sont résolues. Dans ce cas, la solution de F est $f = A(f_1 \ldots f_k)$ les f_i étant les solutions des F_i.

Dans ce dernier cas, f n'est pas un élément initial de la base, mais peut être adjoint à la base (si toutefois il n'existe pas d'empêchement technologique à la réutilisation de f en un autre point du réseau calculé).

Remarque : ce principe permet d'obtenir des réseaux non arborescents, donc a priori plus simple. Toutefois, ceci se fera par une très nette augmentation de la durée du calcul. Si la base initiale se réduit aux

seules variables, la comparaison des fonctions intermédiaires à la base, se fera rapidement. Par contre, si la base comporte en plus des variables, autant d'éléments qu'il y a eu de fractionnements, la comparaison sera plus lente.
On peut alors trouver un compromis et ne pas introduire dans la base toutes les fonctions intermédiaires résolues, mais seulement celles suffisamment simples pour avoir des "chances" de resservir, en prenant par exemple comme critère de simplicité le nombre de monômes ou le nombre de lettres d'une forme minimale.

Exemple :
Nous donnons en premier lieu, pages ci-contre un exemple d'une fonction isolée réalisée par la méthode. A chaque itération, nous avons imprimé les bornes de la fonction fractionnée. Cet exemple totalement inexploitable en pratique permet cependant de suivre pas à pas le fractionnement. Un exemple réel sera donné quelques pages plus loin.

3-1-6 Extensions de la méthode

La méthode de fractionnement peut sans difficulté être étendue à divers opérateurs, voire à plusieurs opérateurs pris simultanément.

3-1-6-1 Opérateur Nand

On étend sans difficulté la méthode à l'opérateur Nand par simple dualisation des fonctions.

3-1-6-2 Opérateurs ET - OU

Dans certaines technologies on réalise des opérateurs ET et OU utilisés alternativement. On pourra refaire la même théorie en utilisant simplement une variante de la représentation des fonctions incomplètes : on écrira :

$$F = [\underline{f}, \overline{f}^{*}]$$

et fractionner F par un opérateur ET revient à découper $(\overline{f})^{*}$, et fractionner F par OU revient à découper $\underline{f}$.

UNE FONCTION F1 EST REALISEE PAR LA METHODE DE FRACTIONNEMENT LITTERALE.
POUR CHAQUE FONCTION INTERMEDIAIRE ON DONNE :
-A- LA BORNE INFERIEURE DE LA FONCTION
-B- LE COMPLEMENT DE LA BORNE SUPERIEURE
-C- LA BORNE INFERIEURE DE LA REDUITE DE LA FONCTION
-D- LA BORNE SUPERIEURE DE CETTE REDUITE.
LE SYBOLE ''NOR'' INDIQUE QUE LA REDUITE EST FRACTIONNEE (EN DESSOUS ET A DROITE DE CE SYMBOLE ON TROUVERA LA LISTE DES SOUS FONCTIONS OBTENUES.)
LA TABULATION INDIQUE LES NIVEAUX DE RECURSIVITE .

F1=

BC'DF' + B'C'D'F' + AC'DE'F' + B'CDE'F +
A'C'D'F' + ACDE'F + A'BDE'F' + A'BC'DE'.

A'BCF + B'C'F + CEF' + ABD'F' + AC'F +
D'F + CD'F' + A'B'DF' + B'DEF' + ACF'.

BC'DF' + B'C'D'F' + AC'DE'F' + B'CDE'F +
A'C'D'F' + ACDE'F + A'BDE'F' + A'BC'DE'.

A'BCF + B'C'F + CEF' + ABD'F' + AC'F +
D'F + CD'F' + A'B'DF' + B'DEF' + ACF'.

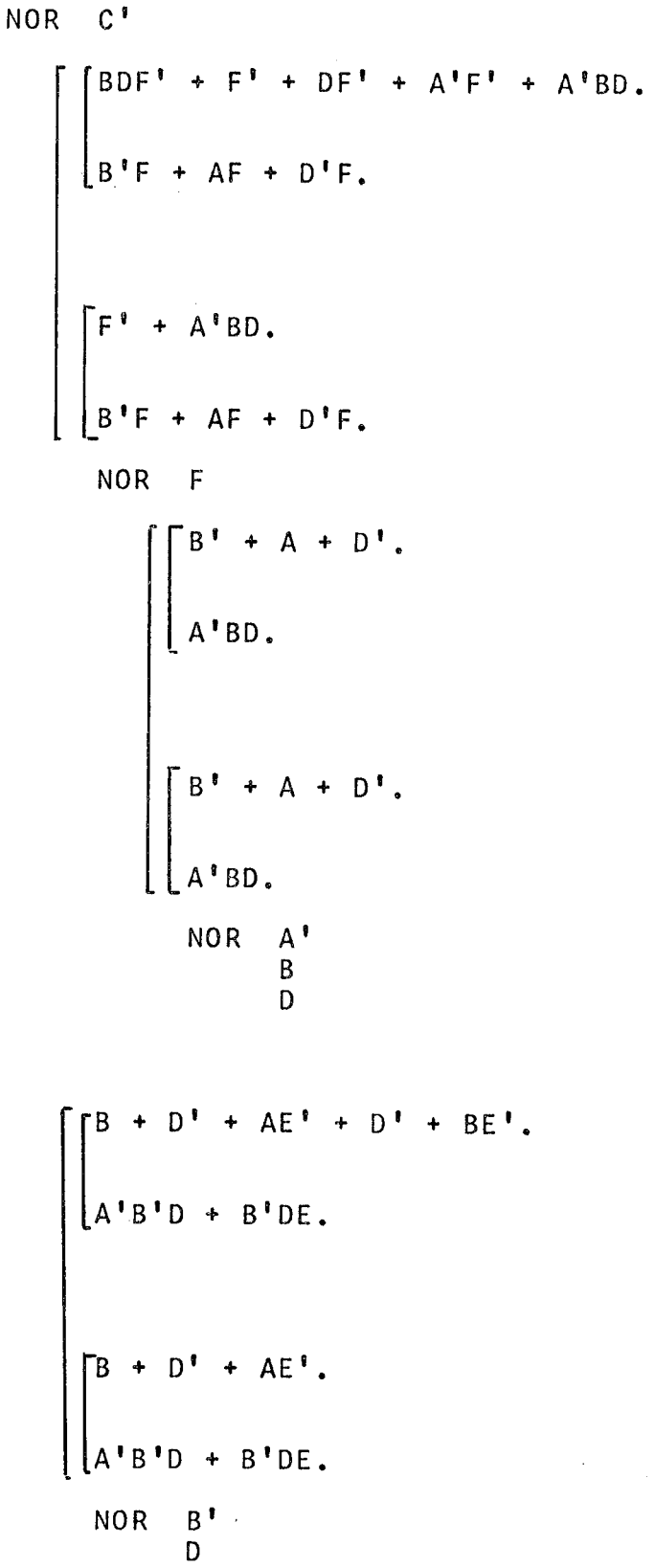
NOR

A'BCF + B'F + CE + ABD' + AF + D'F + CD' +
A'B'D + B'DE + AC.

BC'DF' + B'C'D'F' + AC'DE'F' + A'C'D'F' +
A'BC'DE'.

A'BCF + B'F + CE + ABD' + AF + D'F + CD' +
A'B'D + B'DE + AC.

BC'DF' + B'C'D'F' + AC'DE'F' + A'C'D'F' +
A'BC'DE'.


NOR C'
BDF' + F' + DF' + A'F' + A'BD.
B'F + AF + D'F.
F' + A'BD.
B'F + AF + D'F.
NOR F
B' + A + D'.
A'BD.
B' + A + D'.
A'BD.
NOR A'
B
D
B + D' + AE' + D' + BE'.
A'B'D + B'DE.
B + D' + AE'.
A'B'D + B'DE.
NOR B'
D

```
[[A' + E.
 AE'.

 [A' + E.
  AE'.
NOR  E'
     A
```

```
[[D + B' + D + A' + A'D.
 ABD'.

 [D + B' + A'.
  ABD'.
NOR  D'
     A
     B
```

```
[[A'B + C' + EF' + BD'F' + C' + D' + D'F' +
 A'F' + EF' + F'.
 B'CDE'F + ACDE'F.

 [A'B + C' + D' + F'.
  B'CDE'F + ACDE'F.
 [A'B + C' + D' + F'.
  B'CDF + ACDF.
```

```
NOR   C
      D
      F
      13

⌈⌈F + B'F + E + AD' + AF + D'F + D' + B' +
|| B'E + A.
||
|| A'BDE'F'.
||
||
|| ⌈F + E + D' + B' + A.
||  |
|| ⌊A'BDE'F'.
⌊⌊
  NOR   A'
        E'
        F'
        B
        D
```

EXEMPLE D'UN AUTOMATE SYNCHRONE AYANT :

- 3 ENTREES
- 6 ETATS INTERNES
- 2 SORTIES

SON TABLEAU D'ETAT EST LE SUIVANT,(LA COLONNE DE DROITE CONTENANT LES CODES DES ETATS) :

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

CET AUTOMATE EST REALISE A L'AIDE DE BASCULES D ET DE PORTES NOR A 5 ENTREES

```
F =   NOR  NOR  D'
              NOR  NOR  C
                        NOR  A'
                             B

                   E'

              NOR  B
                   C
                   E

         NOR  F'
              NOR  B'
                   NOR  C'
                        E'
                        A
```

```
                NOR   D'
                      A
                      C

                NOR   E'
                      A
                      C

          NOR   A'
                C'
                B
                D


E =  NOR  NOR   F
                13
                NOR   C'
                      D'
                      A

          NOR   C
                NOR   E'
                      17

                NOR   B'
                      D

                NOR   D'
                      A
                      B

          NOR   C'
                F'
                NOR   E
                      27
                      28

                NOR   E'
                      A
                      D

          NOR   D'
                E'
                F'
```

```
D =    NOR  NOR  C
                 NOR  B
                      22
                      25
                      0

                 NOR  B'
                      F'
                      D

            NOR  D'
                 F'
                 NOR  A
                      NOR  E'
                           B

                      17

                 13

            NOR  B'
                 C'
                 E'
                 F'
                 A
                 D


S1=    NOR  NOR  B'
                 C'
                 17
                 22

            NOR  A'
                 C'
                 B
                 33

            NOR  C'
                 D'
                 E'
                 A
                 B
```

```
S2=    NOR   C'
             NOR   A'
                   NOR   E'
                         B
                         D

             NOR   B
                   F
```

COUT = 39

3-1-6-3 Opérateur majorité.

Bien que cet opérateur ne soit pas privilégié, on peut étendre simplement la méthode.

Soit F fractionné par un OU en F_1, F_2 et F_3. On en déduit un fractionnement par la majorité en posant

$$G_1 = F_1 + F_2,\ G_2 = F_2 + F_3 \text{ et } G_3 = F_1 + F_3 \text{ et alors}$$

$$\begin{aligned}\text{Majorité } (G_1, G_2, G_3) &= (F_1 + F_2)(F_1 + F_3) + (F_1 + F_2)(F_2 + F_3) \\ &\quad + (F_1 + F_3)(F_2 + F_3) \\ &= F_1 + F_2F_3 + F_2 + F_1F_3 + F_3 + F_1F_2 \\ &= F_1 + F_2 + F_3 \\ &= F\end{aligned}$$

La méthode pourrait d'ailleurs s'étendre à tout opérateur à seuil.

3-1-6-4 Opérateurs hybrides NAND-NOR.

On envisage parfois des opérateurs plus complexes tels ceux réalisant la fonction :

$$\begin{aligned}S &= (x'_1 + x'_2)(x'_3 + x'_4).....(x'_{n-1} + x'_n) \\ &= (x_1.x_2)'\ (x_3.x_n)'.....(x_{n-1}\ .\ x_n)'\end{aligned}$$

et on constate aisément qu'il suffit de fractionner une première fois par un opérateur NOR, puis chaque fonction obtenue par un opérateur ET à 2 entrées et à itérer le procédé.

3-1-6-5 Opérateurs privilégiés quelconques.

De la même manière un fractionnement par un opérateur quelconque privilégié pourra toujours se décomposer en fractionnements élémentaires par des opérateurs ET, OU, NAND, NOR.

Exemple.

Page ci-après, nous donnons un exemple complet d'un automate synchrone à trois entrées A, B, C et deux sorties S1 S2 donné par son tableau d'état.

TABLEAU D'ETAT :

	A=0	A=1
0	2,1	7,0
1	3,1	7,1
2	1,1	3,1
3	1,1	7,1
4	3,0	7,0
5	1,0	-,-
6	5,1	4,0
7	7,1	6,1

VARIABLE PRIMAIRE : A
VARIABLES INTERNES : B,C,D.
SORTIE : S

NOTA : CHAQUE BASCULE JK A DEUX ENTREES ELLES MEMES DEDOUBLEES PAR DES ''ET'' FAISANT PARTIE DE LA BASCULE ET NE COMPTANT PAS DANS LE COUT TOTAL.

RESEAU CALCULE:

```
DJ=   ET   A
           NAND B'
                C

DK=   ET   A'
           C'

CJ=   ET   NAND A'
                B
                D

CK=   ET   NAND D'
                A

           NAND B
                D

BJ=   ET   NAND A
                C
                D

           NAND A'
                C'
                D'

BK=   ET   A
           C
           D

S=    NAND NAND NAND D
                     NAND B
                          C
                     NAND A'
                          C
                NAND B'
                     C'
                     A

EOF:
```

cout: 12

temps de calcul: 11 sec.

Le codage, au moyen des variables internes D, E, F est imposé. On dispose pour réaliser cet automate de :

- bascules JK
- portes "ET" à 3 entrées utilisées uniquement aux entrées des JK
- portes NAND à 4 entrées

Le programme détermine les fonctions D1, D2, E1, E2, F1, F2 des entrées des bascules et pour chacune en fait la synthèse. Le résultat est présenté sous une forme très lisible, qui permet de reconstituer instantanément le réseau. Quand une fonction intermédiaire F_i intervient en plusieurs points, elle est désignée par son numéro i.

3-1-7 Conclusion.

La méthode de fractionnement proposée ici s'avère donc très simple d'emploi. Elle a, malgré tout, de graves défauts :

- elle est lente car elle fait appel à un calcul littéral pour lequel les ordinateurs ne sont pas faits
- elle ne fait aucunement intervenir des notions telles que la planéarité des réseaux obtenus, ce qui pourra être un handicap majeur dans des techniques intégrées par exemple.

Nous développerons dans le chapitre suivant une méthode appropriée pour ce type de problèmes.

3 - 2 Méthode topologique.

3-2-1 Introduction.

Les méthodes développées précédemment sont efficaces mais relativement lourdes du fait du calcul littéral effectué sur des polynômes ; elles nécessitent des ordinateurs d'une taille importante, ce dont ne disposent malheureusement pas tous les constructeurs. En outre le problème du dessin des connexions entre portes n'est pas résolu. On est conduit aux classiques problèmes d'implantation traités par exemple par J.SARRET et qui nécessitent également des programmes longs .

Notre idée a été de traiter simultanément le problème de la synthèse et de l'implantation. En effet, il est plus difficile d'implanter un réseau que de le calculer. Il est donc normal de choisir parmi les nombreuses solutions qui s'offrent au cours de la phase synthèse, celle qui donne le réseau le plus simple à implanter.
Nous avons alors abandonné les méthodes habituelles pour envisager des méthodes mieux adaptées au problème posé.
Enfin, il nous a paru nécessaire d'élargir le champ des opérateurs utilisés tels le Nand, le Nor, etc... qui sont simples à construire, mais peu commodes d'emploi. Nous utilisons des opérateurs plus complexes qui présentent des propriétés mathématiques plus intéressantes.

3-2-2 Opérateurs élémentaires.

La méthode que nous proposons dans ce chapitre n'est pas valable pour tout opérateur privilégié, mais seulement pour une catégorie de ces opérateurs que nous appellerons élémentaires.

Définition : nous appellerons opérateur élémentaire, un opérateur $A(x_1, x_2 \ldots\ldots, x_n)$ tel que

- pour toute fonction donnée F,
- pour tout choix de fonctions intermédiaires $y_1, y_2 \ldots\ldots, y_p$ avec

$$1 \leq p \leq n-1,$$

on puisse toujours déterminer $y_{p+1} \ldots\ldots\ldots y_n$ tels que

$$F = A(y_1, y_2 \ldots\ldots, y_p, y_{p+1}, \ldots\ldots, y_n)$$

Cela signifie en pratique qu'il est toujours possible de se servir de $y_1 \ldots\ldots y_p$ pour réaliser F quelles que soient ces fonctions.

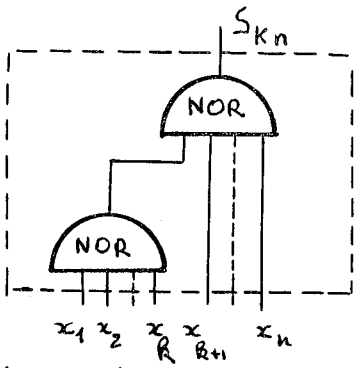
Exemples.

Sont élémentaires les opérateurs réalisant les fonctions suivantes :
$(x_1 + x_2)x_3$, $x_1x_2 + x_3$, $x_1' x_2' + x_3$, $(x_1 + x_2 = x_3)x_4x_5$ etc.....
Ne sont pas élémentaires les opérateurs réalisant les fonctions :

$$x_1 + x_2,\ x_1' x_2' x_3'$$

Remarque 1.

Tous ces opérateurs ne sont pas d'un égal intérêt technologique. Il en est un cependant qui, se construisant avec deux Nor, est très simple



$$S_{kn} = (x_1x_2+\ldots+x_k)x'_{k+1}\ldots x'_n$$

qui est bien élémentaire pour p = k -1

Bien que la méthode soit plus générale, dans tout ce qui suit nous utiliserons uniquement cet opérateur en le limitant même au cas n = 3, k = 2

Remarque 2.

La méthode de fractionnement littérale ne permettait pas une bonne minimisation des réseaux : la probabilité d'obtenir une fonction intermédiaire pouvant servir en plusieurs points du réseau était faible, sauf pour des fonctions simples telles un produit, une somme, un complément de variables. Au contraire, en utilisant la propriété caractéristique des opérateurs élémentaires, nous pourrons utiliser certaines fonctions à notre gré.

3-2-3 Définition du fractionnement.

Etant donnés

- F la fonction à fractionner
- B la base du fractionnement, nous définirons un fractionnement tel qu'au moins un des F_i soit résolu par un élément de la base :

a) on choisira F_1 dans la base

b) on déterminera F_2 et F_3 tels que

$$F = (F_1 + F_2)F'_3 \qquad (1)$$

c) pour F_2 et F_3, on itérera le procédé si toutefois ces fonctions ne sont pas résolues par un élément de la base.

Détermination de F_2 et F_3.

F étant incomplètement spécifiée, peut prendre les 3 valeurs possibles 0, Ø et 1.

F_1 étant 1 élément de la base, est une fonction complètement spécifiée f_1 qui prend les valeurs 0 et 1.

On peut donc dresser un tableau des valeurs à donner à F_2 et F_3 pour vérifier l'équation (1).

F	f_1	F_2	F_3
0	0	$F_2.F'_3 = 0$	
0	1	Ø	1
Ø	0	Ø	Ø
Ø	1	Ø	Ø
1	0	1	0
1	1	Ø	0

On constate qu'il est possible de définir une valeur unique du couple $\{F_2, F_3\}$ sauf pour F = 0 et f_1 = 1 où deux solutions sont possibles :

$F_2 = 0$ et $F_3 = Ø$

ou

$F_2 = Ø$ et $F_3 = 1$

solution qui peut être résumée en

$$F_2 . F'_3 = 0$$

En fait, si pour toutes les n-uplets de valeurs des variables $x_1 \ldots . x_n$ pour lesquels on a $F = 0$ et $f_1 = 0$, on choisissait de prendre $F_2 = \emptyset$ et $F_3 = 1$, on en déduirait les déterminations banales suivantes :

$$F_2 = 1 \text{ et } F_3 = F'$$

ce qui est sans intérêt du point de vue convergence de la méthode .

Expressions de F_2 et F_3.

Le problème du choix de F_2 et F_3 ne peut être résolu de manière rigoureuse. Nous utilisons alors une méthode heuristique. On constate que F_3 est plus souvent spécifié que F_2, donc a priori plus coûteûse à réaliser. Il est normal alors de choisir f_3 solution de F_3 en premier, et ceci étant de calculer F_2 en fonction de F, f_1 et f_3.

Si $F_3 = 0$ alors $F_2 = 0$ sinon $F_2 = \emptyset$

On dresse alors le tableau suivant en détaillant les valeurs de $\underline{f}$ et $(\overline{f})'$

F	f_1	$\underline{f}$	$(\overline{f})'$	F_3	F_2
0	0	0	1	$\emptyset$	$\emptyset . F'_3$
0	1	0	1	1	$\emptyset$
$\emptyset$	0	0	0	$\emptyset$	$\emptyset$
$\emptyset$	1	0	0	$\emptyset$	$\emptyset$
1	0	1	0	0	1
1	1	1	0	0	$\emptyset$

On en déduit les valeurs

$$F_3 = \left[(\overline{f})'.f_1, \underline{f} \right]$$

$$F_2 = \left[\underline{f}.f'_1, \overline{f}'.f'_1.f'_3 \right]$$

3-2-4 Convergence.

Nous cherchons à réaliser le critère défini au paragraphe 3-1-4 : F_3 et F_2 devront être strictement plus incomplètes que F

a) Condition sur F_3

On cherchera à avoir $F_3 \neq F'$. Or $(\overline{f}_3)' = \underline{f}$. Donc on devra définir $\underline{f_3} \neq (\overline{f})'$, c'est-à-dire

$$(\overline{f})'f_1 \neq \underline{f} \qquad (2)$$

b) Condition sur F_2

Au cours de la phase du choix de f_1 on ne connait pas encore f_2. On doit donc imposer $F_2 \neq F$ pour toute détermination possible de F. Donc

$$\underline{f}.f'_1 \neq \underline{f} \qquad (3)$$

Ou bien, quel que soit f_3,

$$(\overline{f})'.f'_1.f'_3 \neq (\overline{f})'$$

Donc, même pour $f'_3 > (\overline{f})'.f'_1$:

$$(\overline{f})'.f'_1 \neq (\overline{f})' \qquad (4)$$

En résumé, on choisira f_1 tel que : ou bien (2) et (3) sont vérifiées ou bien (2) et (4) sont vérifiées.

3-2-5 Choix du meilleur pivot.

La convergence n'est pas suffisante. Nous voulons en plus une rapide convergence. On conçoit intuitivement que si f_1 est voisin de F, alors

F_2 et F_3 seront simples : en effet si f_1 est une solution de F, il existe deux solutions banales pour F_2 et F_3

$$f_2 = 0 \qquad \text{et} \qquad f_3 = 0$$

Nous allons montrer ceci plus rigoureusement.

Propriété. Nous admettons encore le principe suivant : une fonction est d'autant moins coûteuse à réaliser qu'elle est plus incomplète.

Nous allons donc chercher à minimiser les ensembles de points sur lesquels F_2 et F_3 sont déterminés.

Considérant le tableau du paragraphe 3-2-3, on est conduit à distinguer quatre régions.

a) Région caractérisée par $(\overline{f})'.f'_1$. Sur les points de cette région, on a une relative liberté dans le choix de F_2 et F_3. Nous n'imposerons pas à cette région d'être minimale.

b) Région caractérisée par $(\overline{f})'.f_1$. Sur les points de cette région F_3 est déterminé. On minimisera autant que possible le nombre

$$N_3 = |(\overline{f})'.f_1| \text{ de points de cette région.}$$

c) Région caractérisée par $\underline{f}.f'_1$. Sur les points de cette région F_2 est défini. On minimisera le nombre :

$$N_2 = |\underline{f}.f'_1| \text{ de points de cette région.}$$

d) Région caractérisée par $\underline{f}$. Sur les points de cette région F_3 est défini. Mais cette région ne dépend pas du choix de f_1 ; on n'a donc pas à la faire intervenir dans le choix de f_1.

Distance algébrique.

On est conduit à minimiser simultanément N_2 et N_3. On cherchera à minimiser une quantité unique de la forme

$$D_a = \lambda N_2 + \mu N_3$$

Le choix de λ et μ sera arbitraire et défini expérimentalement.

En particulier si $\lambda = \mu = 1$ on a :

$$d_a = |(\overline{f})'.f_1| + |\underline{f}\ f'_1|$$
$$= |(\overline{f})'.f_1 + \underline{f}.f'_1|$$

puisque les deux quantités sont disjointes; donc d est la distance algébrique entre F et f_1.

3-2-6 Méthode expérimentale.

Nous avons vu que le choix de λ et μ est arbitraire. Nous définirons la valeur de λ/μ expérimentalement, en nous aidant des considérations suivantes :

F_3 étant plus souvent spécifié que F_2 est plus complexe : il est normal "d'avantager" F_3 dans le choix de f_1, au détriment de F_2. Nous chercherons à prendre $\lambda/\mu < 1$

Ceci étant, nous avons traité un grand nombre d'exemples de fonctions, et constaté que la meilleure valeur de λ/μ était 0,3 (des différences du simple au double pouvant être constatées dans le coût en nombre d'opérateurs pour de mauvais choix de λ et μ). Cette valeur de λ/μ pourrait éventuellement être influencée par le type de fonctions à réaliser.

3-2-7 On trouvera pages ci-après un exemple calculé par ordinateur.

3-2-8 Généralisations.

1) Opérateur $S_{kn} = (x_1 + \ldots + x_k)x'_{k+1}\ldots\ldots x'_n$

Cet opérateur est élémentaire si on choisit P = k-1. On peut en effet pour tout choix de F et de $f_1, f_2 \ldots f_{k-1}$ trouver $f_k, f_{k+1} \ldots f_n$ telles que $F = (f_1 + \ldots + f_{k-1} + f_k)f'_{K+1}\ldots\ldots\ldots f'_n$

L'extension de la méthode se fait de manière simple : nous énoncerons la méthode sans démonstration.

a) Choix des pivots.

Comme précédemment on choisit pour tout i compris entre 1 et k-1, f_i le plus proche de F.

b) On détermine comme précédemment la quantité

$$G = \sum_{i=k+1}^{n} F_i$$

c) La fonction G sera alors fractionnée en somme par exemple de manière aléatoire : les points de G étant répartis au hasard entre les diverses bornes inférieures f_i.

2) Opérateurs élémentaires quelconques.

Ils se déduisent de l'opérateur précédent par dualisation ou complémentation de certaines fonctions intermédiaires, ce qui se fera sans aucune difficulté avant chaque itération.

3 - 3 Synthèse et implantation simultanée.

Introduction.

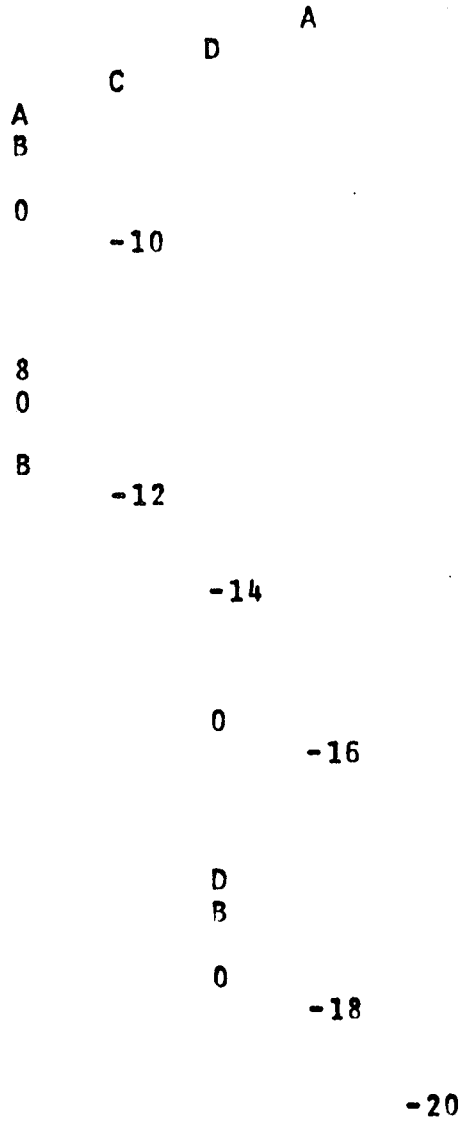
La méthode exposée précédemment fait intervenir une topologie de l'espace des fonctions. A ce titre elle sera aisément utilisable avec une méthode faisant intervenir une topologie dans la réalisation pratique du circuit.

Cette méthode sera utilisable essentiellement en intégration à grande échelle où les règles d'implantation et de cablage sont très strictes

EXEMPLE : SYNTHESE DE DEUX FONCTION PAR LA METHODE TOPOLOGIQUE .
CHAQUE OPERATEUR EST CONSTITUE DE DEUX NOR EN SERIE ET EST SYMBOLISE PAR LE NOMBRE -I , I ETANT LE NUMERO DU DEUXIEME NOR,ET I-1 ETANT LE NUMERO DU PREMIER.
LE FAN IN DES OPERATEURS NOR EST DE 2 .
LES COMPLEMENTS DES VARIABLES A,B,...F SONT REPRESENTES PAR LES NOMBRES 9,10,....14 .
LES CONNEXIONS VERS DES PORTES SERVANT PLUSIEURS FOIS SONT INDIQUEES PAR LE NUMERO DE CES PORTES .

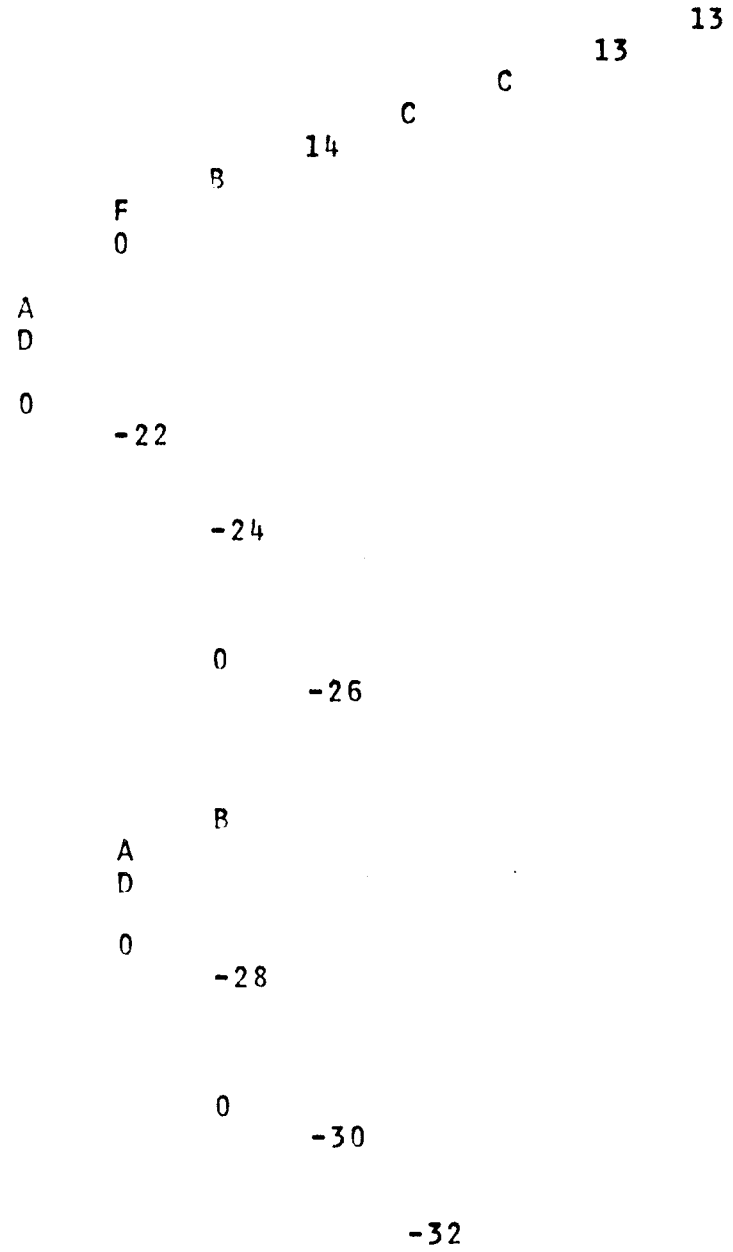
AC'D'F' + AD'E'F' + A'BDE' + AD'EF + ACD'F + AB'DE'.

A'C'D'F' + A'D'E'F' + ABDE' + A'D'EF + A'CD'F + A'B'DE'.



B'C'DE'F + BCD'E'F + A'BCE'F + ABDE'F' + A'C'DE'F + B'C'D'EF + BC'D'EF' +
B'CE'F'.

CD'EF + B'C'E'F' + B'CE'F + BD'EF + ABDE'F + A'BE'F' + BD'E'F' + B'C'D'F'.



-32

0

-34

D

14

31

0

0

-36

A

B

0

-38

-40

0

-42

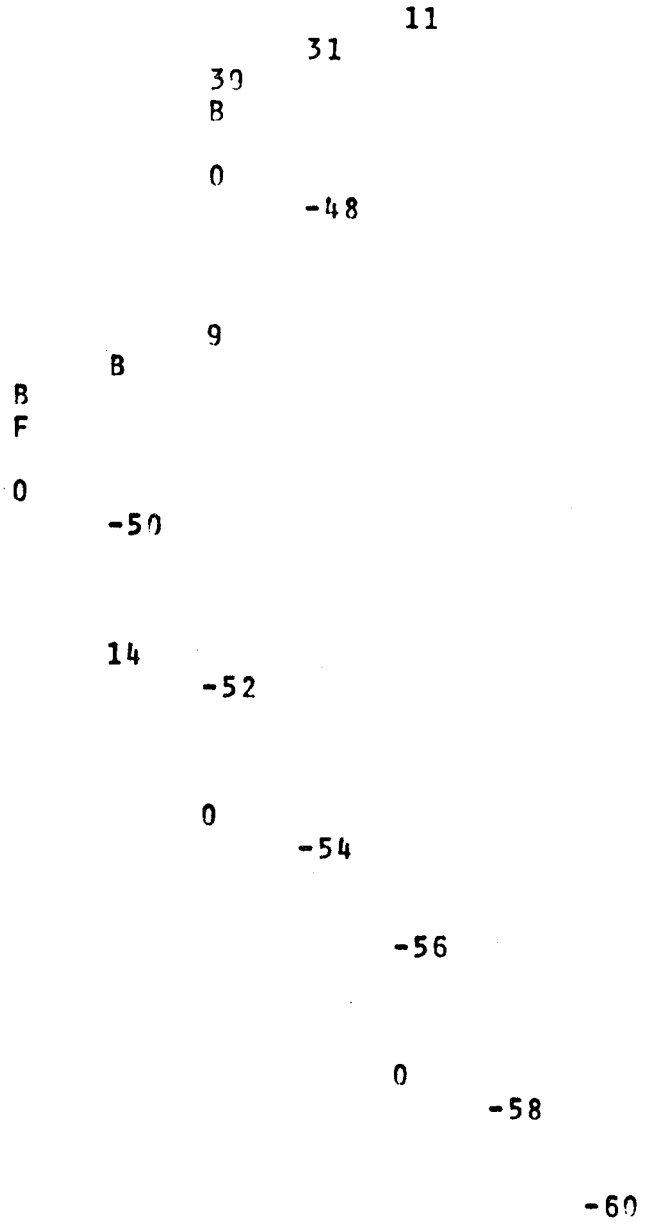
-44

0

-46

11

31


11
31
39
B
0
-48
9
B
B
F
0
-50
14
-52
0
-54
-56
0
-58
-60

et peuvent s'interpréter mathématiquement.
Enfin, précisons que les calculs de synthèse, d'implantation et de cablage se feront simultanément et non en trois étapes comme dans les méthodes habituelles.

3-3-1 Structures.

1) Dispositions.

Nous imposerons aux circuits intégrés utilisés d'avoir une structure géométrique définie et répétitive.
Les plus utilisées de ces structures sont :
a) la structure dite matricielle où les opérateurs sont disposés suivant les noeuds d'un quadrillage
b) la structure dite en ligne, telle celle utilisée au L E T I et appelée structure d'"Espalier", où les opérateurs sont sur une ou deux lignes.
Dans le cas de cablage sélectif, nous pourrons admettre des défauts dans la structure : opérateurs qui manquent à certains noeuds, ou dans toute une zone.

2) Connexions.

Les connexions qu'elles soient sur une même puce ou vers l'extérieur, devront, elles aussi, avoir une structure géométrique simple : les connexions vers l'extérieur se feront par des "plots de contact" régulièrement répartis sur la périphérie de la puce.
Les connexions entre éléments se feront suivant des règles précises et simples : à chaque phase du calcul du réseau, et pour une nouvelle connexion à rajouter à la partie du réseau déjà construite, on devra pouvoir définir un coût fonction de paramètres tels que la longueur de la connexion, le nombre de croisements nécessaires, le nombre de changements de direction etc.....

Définition : on appellera distance technologique entre deux points, le coût de la connexion entre ces deux points.

3-3-2 Principe.

L'algorithme de synthèse, implantation et cablage simultanés, se décompose, pour chaque fonction à traiter, en plusieurs phases.

a) Choix du point précis A où cette fonction devra être réalisée. Cette phase interviendra en particulier pour les fonctions devant être "sorties" de la puce : on devra faire choix d'un plot de contact. Ce choix se fera en général au hasard ; on prendra en général le premier des plots disponibles, ceux-ci ayant été numérotés au préalable.

b) Choix de l'emplacement de l'opérateur devant construire la fonction. On choisira cet opérateur en fonction du coût de la connexion AB à établir entre sa sortie B et le point A défini précédemment. En général on cherchera l'opérateur disponible le plus proche de A au sens de la distance technologique.

c) Fractionnement. Ce fractionnement se faisant par la méthode topologique consistera essentiellement en un choix correct des pivots. On cherchera à minimiser :

- la distance algébrique d_a entre F à réaliser et le pivot choisi f_1

- la distance technologique d_t entre le point d'entrée C de f_1 dans l'opérateur réalisant F, et le point D où f_1 est disponible.

En pratique, on minimisera une quantité

$$D = p.da + q.dt$$

les valeurs optimales à donner à p et q étant définies expérimentalement pour chaque structure.

Remarque : le point P où f_1 est disponible peut n'être pas connu : si f_1 est une fonction de la base non encore présente sur la puce (une variable par exemple) on définira alors P comme le plot de sortie disponible le plus proche de C.

d) Itération. Le fractionnement étant défini, on recherche

pour chaque fonction intermédiaire F_i obtenue, s'il existe une solution f_i dans la base.
Cette recherche des solutions se fera en tenant compte de la condition de compatibilité, mais aussi de la possibilité d'effectuer pratiquement la connexion.
On pourra, par exemple, se fixer une borne supérieure C_{max} du coût d'une connexion. Si on trouve dans le réseau déjà câblé une fonction f_i solution de F_i, on pourra rejeter cette solution si le coût de la connexion est supérieur à C_{max}. (Nous ne rentrerons pas ici dans les détails de cette partie, les considérations technologiques nous entraînent trop loin).
Pour tout F_i n'ayant pas de solution satisfaisante, on itère le procédé en (a).

3-3-3 Remarques diverses.

(1) La méthode se prête à toutes les structures, pourvu qu'il soit possible de définir le coût d'une connexion de manière rigoureuse. En particulier, il a été possible d'appliquer cette méthode à la conception dans le cas de structures d'espalier

(2) La méthode se prête bien au calcul de réseaux avec cablage sélectif. Nous avons vu en effet apparaître une notion de disponibilité d'opérateurs : un opérateur peut être indisponible parce qu'il est déjà utilisé dans une autre partie de réseau, ou parce qu'il est "en panne". Introduire une carte des opérateurs en bon état revient donc à initialiser de manière convenable, le tableau des opérateurs disponibles.

CHAPITRE 4

ALEAS

4 - 1 Rappel et définitions.

4-1-1 Fonctions incomplètes.

S'il n'est pas nécessaire de tenir compte des causes de l'indétermination pour le calcul des réseaux, ceci est cependant indispensable dès qu'on étudie des phénomènes physiques tels que les aléas.
On admet en général qu'une fonction incomplète F(X) peut être non spécifiée en un point X_o pour trois raisons principales.

a) Dépendance des variables.

Si les variables x_i, x_2,.....x_n ne sont pas indépendantes, certaines combinaisons de leurs valeurs sont impossibles. Donc la valeur de F n'a pas à être définie sur ces valeurs.

b) Indétermination réelle.

Il peut se produire qu'on ne se soucie pas de la valeur que prendra F pour certaines valeurs des variables.
Par exemple, un automate destiné à assurer la mise à feu d'une bombe n'aura pas besoin d'être spécifié après délivrance du signal de mise à feu, puisqu'il est alors détruit. Ce type d'indétermination sera très différent, du point de vue des aléas, du type C.

c) Indétermination par composition.

Si F n'est pas utilisée en tant que telle, mais combinée avec d'autres

fonctions, elle n'a pas en général à être définie partout. Par exemple, dans un fractionnement de G en G = F + H, F n'a pas à être spécifiée sur les points où H = 1.

4-1-2 Aléa statique et dynamique.

Définition 1 : un aléa statique sur f(X) est une transition entre deux valeurs d'entrées adjacentes X_o et X_1 telles que $f(X_o) = f(X_1) = \alpha$, transition au cours de laquelle pourra apparaître une fausse valeur momentanée $f = \alpha'$.

Définition 2 : un aléa dynamique est une transition entre deux valeurs adjacentes d'entrées X_o et X_1 telles que $f(X_o) = \alpha$, $f(X_1) = \alpha'$, transition au cours de laquelle f peut prendre successivement les valeurs $\alpha, \alpha', \alpha, \alpha'$.

4-1-3 Classification des aléas.

Soit un aléa sur la transition $X_o \rightarrow X_1$ des variables d'entrées, trois cas peuvent être envisagés suivant que $F(X_o)$ et $F(X_1)$ sont ou non spécifiés.

1) $F(X_o)$ et $F(X_1)$ sont spécifiés tous deux ; les aléas, qu'ils soient statiques ou dynamiques doivent être corrigés. On dira ici que les aléas sont graves.

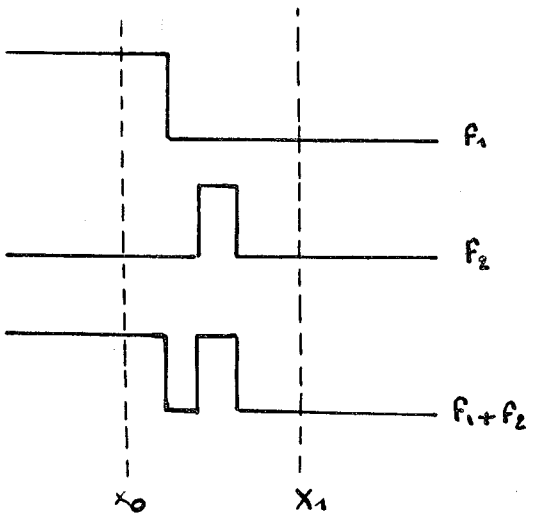
2) $F(X_o)$ et $F(X_1)$ sont non spécifiés tous deux. Quelqu'en soit la raison l'aléa est ici sans importance.

3) $F(X_o)$ est non spécifié et $F(X_1)$ est spécifié (ou l'inverse). Il y a ici trois cas à distinguer suivant la cause de la non spécification.

a) Si $f(X_o)$ est une indétermination de type (a) (par dépendance des variables), la transition $X_o \rightarrow X_1$ est impossible et l'aléa n'a pas d'importance.

b) Si $f(X_o)$ est une indétermination de type (b), la valeur de f étant réellement sans importance sur X_o, on peut en général admettre que les aléas au voisinage de X_o sont sans importance.

c) Si $f(X_o)$ est une indétermination par composition (de type (c)), il est nécessaire d'empêcher la présence d'aléas, en particulier d'aléas statiques, qui pourront par composition créer un aléa dynamique comme le montre le diagramme ci-dessous :



il y a un aléa statique sur f_2

il y a donc un aléa dynamique sur $f_1 + f_2$

4 - 2 Correction des aléas.

Nous cherchons à développer une méthode de synthèse par fractionnement donnant des réseaux sans aléas. Nous limitons nos recherches aux méthodes de fractionnements littérales développées en 3-1.
Nous étudions uniquement le fractionnement en sommes auquel peut se ramener tout fractionnement par des opérateurs tels que ET, NOR, NAND.

4-2-1 Aléa dans un fractionnement.

Définitions.

Soit F fractionné en $\sum_{i=1}^{p} F_i$ et soient $f_1, f_2 \ldots . f_p$ les déterminations choisies pour chacun des F_i. Nous dirons que le fractionnement introduit un aléa statique s'il existe une transition $X_o \longrightarrow X_1$ telle que :

$$\exists f_i,\ f_i(X_o) = 0 \quad \text{et} \quad f_i(X_1) = 1$$
$$\exists f_j,\ f_j(X_1) = 1 \quad \text{et} \quad f_j(X_1) = 0$$
$$\nexists f_k,\ f_k(X_o) = f_k(X_1) = 1$$

Nous dirons que le fractionnement n'introduit pas d'aléa s'il n'existe pas un tel couple (X_o, X_1). Cela ne signifie pas que F est sans aléa statique, car ceux-ci peuvent provenir d'un quelconque des f_i.

Aléa dynamique introduit par un fractionnement.

Nous dirons qu'un aléa dynamique est introduit par le fractionnement de F en $\sum_{i=1}^{P} F_i$ s'il existe une transition $X_o \rightarrow X_1$ telle que f_i ayant un aléa de type 1 au cours de la transition $X_o \rightarrow X_1$

$$\exists f_j \ , \ f_j(X_o) = 0 \quad \text{et} \quad f_j(X_1) = 1$$

$$\nexists f_k \ , \ f_k(X_o) = 1 \qquad f_k(X_1) = 1$$

Nous nous intéressons en fait aux seuls aléas dynamiques qui ne seront pas masqués par un aléa statique, c'est-à-dire tels qu'il n'existe pas f_m avec $f_m(X_o) = 1$ et $f_m(X_1) = 0$ (il y aurait alors un aléa statique entre f_m et f_j, cet aléa devant être corrigé, l'aléa dynamique le sera du même coup).

Comme précédemment nous distinguons ce cas de celui où l'aléa dynamique est introduit par l'un des f_i.

Propriété des aléas statiques.

Dans un fractionnement en somme, on ne peut introduire que des aléas de type 0 (alors que chacun des f_i peut introduire un aléa quelconque).

Propriété.

Tout aléa est soit introduit par le fractionnement soit introduit par un des f_i. Ceci découle de manière évidente des définitions.

4-2-2 Définition d'un fractionnement correct.

Théorème 1.

Soit F une fonction à réaliser, et B la base du fractionnement conte-

nant des fonctions déjà réalisées sans aléa.

Si, à chaque étape le fractionnement n'introduit pas d'aléa statique, alors F est sans aléa statique.

Constatons au préalable que toute fonction intermédiaire est définie comme une combinaison de fonctions de la base (fonction de la base initiale, ou introduite au cours de la méthode). Il suffira alors pour démontrer le théorème, de montrer que toute nouvelle fonction introduite dans la base est sans aléa.

Ceci se démontre par récurrence :

a) au préalable la base est sans aléa statique (étant formée de variables)

b) supposons qu'à une étape donnée la base B_p soit formée de $f_1, f_2 \ldots\ldots f_p$ sans aléa statique. Montrons que la nouvelle fonction introduite :

$$f_{p+1} = \sum_{i \in I} f_i$$ est sans aléa statique. Par définition, f_{p+1} est une solution particulière de

$$F_{p+1} = \sum_{i \in I} F_i$$

Si, par hypothèse le fractionnement n'introduit pas d'aléa statique quelle que soit le détermination f_i choisie pour chaque F_i, alors ceci est vrai en particulier quand chaque f_i est une fonction de la base. D'après la propriété du paragraphe 4-2-1, f_{p+1} est donc sans aléa statique, donc la nouvelle base B_{p+1} est encore sans aléa.

Lemme.

Pour que $f = \Sigma f_i$, (où chacun des f_i est sans aléa), soit sans aléa, il faut et il suffit que pour tout couple $\{X_o, X_1\}$ de points voisins tels que

$$f(X_o) = f(X_1) = 1$$

il existe au moins un f_i tel que

$$f_i(X_o) = f_i(X_1) = 1$$

a) La condition est manifestement suffisante : f_i ainsi définie, masque toute fausse valeur dans la transition entre X_0 et X_1.

b) La condition est nécessaire : s'il n'existe pas un tel f_i, alors il existe au moins un f_j tel que

$$f_j(X_o) = 1 \text{ et donc } f_j(X_1) = 0$$

et il existe au moins un f_k tel que

$$f_k(X_1) = 1 \text{ et donc } f_k(X_o) = 0$$

et il y a introduction d'un aléa.

Corollaire.

Si à tout couple $\{X_o, X_1\}$ de points de f on peut associer un f_i tel que $f_i(X_o) = f_i(X_1) = 1$, alors a fortiori la propriété reste vraie si on remplace f_i par une fonction plus grande.
Si donc

$$\underline{f} = \Sigma \underline{f_i} \text{ est sans aléa, a fortiori :}$$

$$f = \Sigma f_i \text{ avec } f_i > \underline{f_i}$$

est sans aléa.

Théorème 2.

Si le fractionnement se fait de manière à ce que toutes les fois qu'il existe un monôme m_1x dans $\underline{f_i}$ et m_2x' dans $\underline{f_j}$, on trouve dans un quelconque des $\underline{f_k}$ le monôme consensus $m_1.m_2$ (ou un monôme plus grand), la condition de suppression des aléas sur $\underline{f}$ est vérifiée, donc quels que soient les f_i choisis, il n'y aura pas d'aléa sur f.

Remarque.
La condition n'est pas nécessaire.
Exemple : $f_1 = ax$ $f_2 = bx'$ $f_3 = ay + by'$

f_3 n'est pas le consensus de f_1 et f_2 mais il corrige l'aléa entre f_1 et f_2.

Corollaire.

Si le fractionnement s'opère sur une base première complète de f, la condition ci-dessus est remplie, et il n'y aura pas d'aléa grave sur F.

Remarque.

Nous avons montré l'absence d'aléa grave. Par contre, si, par exemple F est non spécifiée sur X_o il pourra se produire un aléa statique sur F sur la transition $X_o \longrightarrow X_1$, aléa qui pourra donner lieu à un aléa dynamique grave sur une fonction telle que :

$$G = H.F$$

$$\text{avec} \quad H(X_o) = 1,\ H(X_1) = 0$$

4-2-3 Aléa dynamique.

Théorème 3.

Etant donnée F une fonction à réaliser et B la base initiale contenant au départ des fonctions sans aléa.
Si, à chaque étape le fractionnement n'introduit pas d'aléa dynamique, alors F est sans aléa dynamique.
La démonstration se fait de la même manière que pour le théorème 1.

Application.

Nous avons vu qu'un aléa dynamique peut être introduit dans un fractionnement de F en $\sum_{i=1}^{P} F_i$ si

$$\exists\ f_i \text{ ayant un aléa statique}$$

$$\exists\ f_j \quad f_j(X_o) = 0 \quad f_j(X_1) = 1$$

$$\nexists\ f_k \quad f_k(X_o) = 1 \text{ et } f_k(X_1) = 1$$

et par ailleurs nous avons montré que f_i peut avoir un aléa statique si f_j étant égal à 1 en X_1, on en déduit que f_i n'a pas à être spécifié en X_1.

Conséquence.

Pour interdire la présence d'un aléa dynamique, il suffit d'interdire la présence d'un aléa statique sur la frontière de f_j. Autrement dit, il suffit d'interdire à f_i d'être non spécifiée sur la frontière de f_j.

Corollaire.

Un fractionnement correct de $F = \{\underline{f}, \overline{f}\}$ pourra s'obtenir de la manière suivante :

a) les bornes inférieures des F_i sont déterminées de manière à éliminer les aléas statiques graves. On fractionnera une base première complète de $\underline{f}$.

b) On élimine les aléas dynamiques par le choix des $\overline{f_i}$ qui seront définis par la formule

$$\overline{f_i} = \underline{f_i} + \sum_{j \neq i} \text{Int}(\underline{f_j})$$

Exemple.

(1) Nous allons montrer un exemple de fonction réalisée par un réseau dans lequel il n'y a nulle part d'aléa statique grave, mais où il y a un aléa dynamique grave.

Soit à réaliser la fonction

$$F = \left[xy'z' + yz,\ x'y' + y'z + yz' + x'z'\right]$$

Respectons les règles pour obtenir un réseau sans aléa statique : nous fractionnons F en $F = NI(F_1, F_2)$ avec

$$F_1 = \left[y'z,\ xy'z' + yz\right]$$

dont une solution est $f = y'z$

$$F_2 = \left[x'y' + yz' + x'z',\ xy'z' + yz\right]$$

La borne inférieure $yz + xy'z'$ de F'_2 est bien une base première complète donc son fractionnement donnera pour toute solution de F_2 un réseau sans aléa statique grave. Nous fractionnons par exemple F_2 en :

$$F_3 = [xy'z',\ x'y' + yz' + x'z']$$

$$F_4 = [yz,\ x'y' + yz' + x'z']$$

F_3 a une solution simple $f_3 = xy'$

F_4 a une solution simple $f_4 = yz$

On obtient donc pour F la solution :

$$f = NI(f_1, NI(f_3, f_4))$$

$$= (y + z')(xy' + yz)$$

Il n'y a pas sur f d'aléa statique, mais il y a un <u>aléa dynamique</u> au cours de la transition suivante

$$(x = 1,\ y = 0,\ z = 1) \longrightarrow (x = 1,\ y = 1,\ z = 1)$$

En effet, sur F_2 il y a un aléa statique (qui ne peut être grave par construction) sur cette transition. Montrons ceci sur un tableau de Karnaught.

F_2'

0	1
ϕ	ϕ
1	1
0	0

F_3

0	1
ϕ	ϕ
ϕ	ϕ
0	0

F_4

0	ϕ
ϕ	ϕ
1	1
0	0

f_3

0	1
0	1
0	0
0	0

f_4

0	0
0	0
1	1
0	0

et on a créé ainsi un aléa qui n'est pas grave pour F_2.
Mais cet aléa est à la frontière de F_1 et donne par composition un aléa dynamique sur F.

(2) Etude d'une bonne solution.

L'erreur dans la méthode ci-dessus est de permettre à F_2 d'être non spécifiée à la frontière de F_1 : la borne supérieure $\overline{f_2}$ de F_2 doit être définie par

$$\overline{f_2} = \underline{f_2} + \mathrm{Int}(\underline{f_1})$$

$$= \underline{f_2} \text{ puisque cet intérieur ici est nul,}$$

on a donc

$$F_2 = \left[x'y' + yz' + x'z',\ xy' + yz\right]$$

Si alors, on désire fractionner F_2, il faut faire ceci sur une base première complète de $(\overline{f_2})'$

$$F_2 = \left[x'y' + yz' + x'z',\ xy' + yz + xz\right]$$

Le monôme xz doit être rajouté et le fractionnement de F_2 donnera par exemple

$$\underline{f_3} = xy' \qquad \overline{f_3} = \underline{f_3} + \mathrm{Int}(yz + xz) = \underline{f_3}$$

$$f_4 = yz + xz \qquad \overline{f_4} = \underline{f_4} + \mathrm{Int}(xy') = \underline{f_4}$$

Les deux fonctions F_3 et F_4 sont ici complètes et donnent comme seules solutions :

$$f_3 = xy' \qquad f_4 = yz + xz$$

et, en définitive :

$$f = (y + z')(xy' + yz + xz)$$

fonction qui est cette fois sans aléa.

C H A P I T R E 5

TEST DES SYSTEMES COMBINATOIRES

La méthode de fractionnement littérale permet d'aborder le problème du test des réseaux. L'idée directrice de ce chapitre est de définir une méthode de synthèse de réseaux tels que le calcul de leur séquence de test soit réduit.

On constatera en fait que l'avantage de cette idée est double :

- d'une part le prix du test sera réduit puisqu'il ne sera éventuellement plus nécessaire de calculer les séquences de test sur ordinateur

- d'autre part en éliminant en cours de calcul de nombreux réseaux non testables, on diminue le nombre de solutions possibles, donc le temps de calcul dans la partie synthèse.

5 - 1 Considérations technologiques.

En pratique, il est préférable de vérifier le bon fonctionnement d'un réseau sur des transitions et non sur des valeurs établies, ceci pour deux raisons.

La première, évidente, est qu'il peut être utile de vérifier le fonctionnement en régime transitoire, par exemple pour contrôler les temps de commutation.

La deuxième raison est que les pannes ne sont pas forcément franches : en particulier dans le cas d'éléments à seuil, on pourra observer des vieillissements d'éléments qui modifient le seuil sans pour autant provoquer le "collage" à 1 ou 0.

Par exemple, une porte réalisant Maj(x, y, z) pourra, à cause d'un vieillissement, être transformée en x + y + z ou x.y.z

Si alors on teste, pour les valeurs voisines (x=0 y=0 z=1) et (x=0, y=1, z=1), on peut déceler le changement de seuil, alors que le test sur deux points tels que (x=0, y=0, z=0) et (x=1, y=1, z=1) ne permet pas de déceler les deux pannes ci-dessus.

5 - 2 Application à la synthèse par fractionnement.

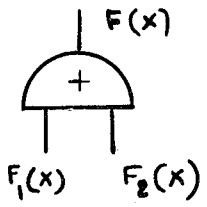
Nous allons chercher les conditions pour qu'une fonction soit testable sur sa frontière. Soit F(X) une fonction incomplète à réaliser. Nous utilisons une méthode de fractionnement pour le calcul du réseau réalisant F :

1) On considère la porte ultime du réseau, celle dont la sortie est F. Cette porte est testable sur la frontière de F.

2) Si F est fractionnée en k sous fonctions plus simples $F_1 \ldots\ldots F_k$, chacune étant réalisée par un réseau dont la porte ultime est $P_1 \ldots\ldots P_k$, on doit pouvoir tester $P_1 \ldots P_k$ sur la frontière de $F_1 \ldots F_k$ c'est-à-dire que les variations d'un des F_i ne doivent pas être "masquées" par les autres.

3) On itère les contraintes du (2) en fractionnant chacun des F_i.

Cas d'une porte OU.



Considérons le cas de la porte OU à deux entrées, cas auquel on pourra ramener la porte OU à k entrées et celui des portes ET, NI etc.... (Cf paragraphe 5-4-1)

Soit F une fonction incomplète à fractionner en somme :

$$F(X) = F_1(X) + F_2(X)$$

Nous envisageons d'abord le cas où F, F_1 et F_2 sont des fonctions complètes (on trouvera la généralisation au paragraphe 5-4-2). Donc étant donnée une fonction complète f, on cherche f_1 et f_2 telles que :

a) $f = f_1 + f_2$

b) f_1 et f_2 soient testables par transition.

Théorème.

Pour que f_1 soit testable par transition, il faut et il suffit qu'il existe deux points voisins X_o et X_1 tels que

a) $f_1(X_o) = 0$ et $f_1(X_1) = 1$

b) $f_2(X_o) = 0$ et $f_2(X_1) = 0$ pour que f_2 ne masque pas les transitions de f_1. Ceci peut également s'exprimer en disant que

$$\exists\, x_i,\ \mathrm{Var}_{x_i}(f_1) \not\subseteq \mathrm{Adher}_{x_i}(f_2)$$

Corollaire : condition suffisante.

S'il existe x_i, tel que f_1 et $\mathrm{Adher}_{x_i} f_2$ sont disjoints, alors f_1 est testable, puisqu'alors la frontière de f_1 n'est manifestement pas incluse dans $\mathrm{Adher}(f_2)$.

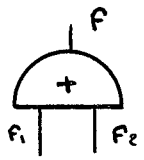
Condition :

$$f_1 . \text{Adher}(f_2) = 0$$

Exemple.

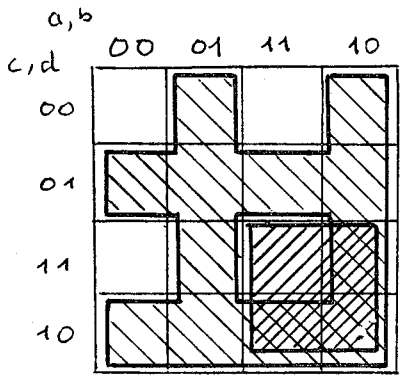
Soit f(abcd) = ab' + ba' + cd' + c'd + ac

et réalisons cette fonction par le réseau



$f_1 = ac$

$f_2 = ab' + ba' + cd' + c'd$



f_1 :

f_2 :

On constate que toutes les transitions de f_1 sont masquées par f_2, (bien que f_1 ne soit pas redondant)

Ce fractionnement n'est pas intéressant. Ceci peut se voir sur les frontières et adhérences de f_1 et f_2 :

$\text{Var}_a(f_1) = c$	$\text{Adher}_a(f_2) = 1$
$\text{Var}_b(f_1) = 0$	$\text{Adher}_b(f_2) = 1$
$\text{Var}_c(f_1) = a$	$\text{Adher}_c(f_2) = 1$
$\text{Var}_d(f_2) = 0$	$\text{Adher}_d(f_2) = 1$

Par contre, si on fractionne f de manière différente en posant

$f_1 = ac + cd'$

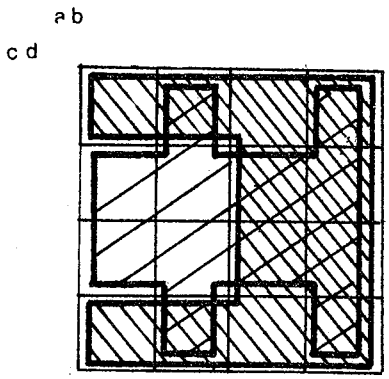
$f_2 = ab' + ba' + c'd$

et on obtient

$$\underset{c}{\text{Var}}(f_1) = a + d'$$

$$\underset{c}{\text{Adher}}(f_2) = ab' + ba' + d$$

et la première expression n'est pas incluse dans la deuxième. On peut donc tester f_1 par transitions. Représentons ceci par le diagramme suivant :



$\text{Var}(f_1)$

$\text{Adher}(f_2)$

les deux transitions indiquées par des flèches sont des transitions dans $\text{Var}(f_1)$ et non dans $\text{Adher}(f_2)$ donc permettent de tester f_1.

5 - 3 Algorithmes de fractionnements.

5-3-1 Définition.

Un fractionnement de f en $f_1 + f_2$ sera dit "bon pour f_1" s'il existe x_i tel que

a) $\left[f_1(x_i) + f_1(x'_i) \right] . f_2 = \underset{x_i}{\text{Adher}}(f_1) . f_2 = 0$

b) f_1 dépend effectivement de x_i, ou encore :

$$f_1(x_i) \oplus f_1(x'_i) \neq 0$$

5-3-2 Propriétés

(1) Tout fractionnement bon pour f_1 est également bon pour f_2 avec le même x .

En effet, posons : $f_1 = ax + bx'$ et $f_2 = cx + dx'$

et supposons : $\underset{x}{\text{Adher}}(f_1).f_2 = 0$ c'est-à-dire

$$(a + b)(cx + dx') = 0$$

pour x = 1 on a donc : $(a + b)c = 0$

pour x = 0 on a : $(a + b)d = 0$

et donc : $(a + b)(c + d) = 0$

et à fortiori : $(ax + bx')(c + d) = 0$

c'est-à-dire : $f_1 \cdot \underset{x}{\text{Adher}}\ (f_2) = 0$

Remarque

Géométriquement la condition $f_1 \cdot \text{Adher}\ f_2 = 0$ signifie que f_1 et f_2 sont à distance 2 c'est-à-dire que tout monome m_i de f_1 et tout monome n_j de f_2 diffèrent par au moins deux variables.

(2) fractionnement suivant une variable

On a vu au chapitre 3 qu'il est intéressant de fractionner une fraction :

$$f = Ax + Bx' + C$$

en $f_1 = Ax$ et $f_2 = Bx' + C$

étudions dans quelles conditions ce fractionnement est bon pour tout $x_i \neq x$:

$$f_1 \cdot \underset{x_i}{\text{Adher}}(f_2) = 0 \quad \text{est équivalent à :}$$

$$Ax \cdot (x'(\underset{x_i}{\text{adher}}(B)) + \underset{x_i}{\text{Adher}}(C)) = 0$$

on a encore :

$$Ax \cdot \underset{x_i}{\text{Adher}}(C) = 0$$

ou encore puisque A et C ne dépendent pas de x :

$$A \cdot \underset{x_i}{\text{Adher}}(C) = 0$$

(autrement dit on doit pouvoir tester f_1 quand x = 1).

On peut donner alors un algorithme de construction de la forme $f = Ax + Bx' + C$ donnant un bon fonctionnement :

- a - On choisit au départ une base première irredondante de f :

$$f = A_o x + B_o x' + C_o$$

- b - Chaque monôme m_i de C_o est comparé à tous les monômes n_j de A_o. S'il existe n_j de A_o qui ne diffère de m_i que d'une variable, m_i est décomposé en deux :

$$m_i = m_i x + m_i x'$$

$m_i x$ étant rajouté à $A_o x$ pour former $A_1 x$
$m_i x$ étant rajouté à $B_o x$ pour former $B_1 x'$
et on itère le procédé pour former sur la base

$$A_1 x + B_1 x' + C_1$$

Le nombre de monômes de C_o étant fini on arrivera en définitive à une forme permettant un bon fonctionnement.

Remarques - Il se peut que le C final soit nul, si au départ, par exemple, chaque monôme de C_o était voisin d'un monôme de A.

- La méthode peut évidemment être simplifiée si on impose à la base de départ d'être canonique :

Si $f = Ax + Bx'$ le fractionnement en $f_1 = Ax$ et $f_2 = Bx'$ sera évidemment bon pour tout $x_i \neq x$

Définition - Un tel fractionnement sera dit uniformément bon

5-3-3 Fractionnement final

Définition Un fractionnement de f en $f_1 + f_2$ sera dit "final" si f_1 ou f_2 est une variable.

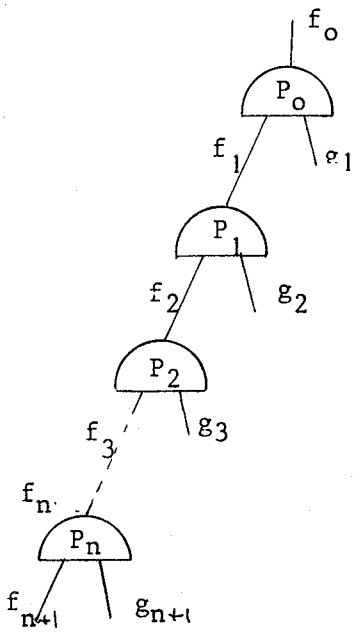
Propriété Si un fractionnement de f est final, la fonction f_i, si elle existe, qui n'est pas une variable peut être testée par transition sur la sortie de la porte effectuant le fractionnement. Si par exemple $f_1 = x$:

$$f = x + f_2(x_1\ x_2 \ldots . x_n) \qquad x \quad x_1 \ldots . x_n$$

si on donne a x la valeur zéro, f_2 peut être testée sur les transitions de l'un quelconque des x_i en observant les variations de f.

5-3-6 Théorème

Soit dans un réseau quelconque (arborescent ou non), une chaîne $P_o\ P_1\ P_2 \ldots . P_n$ de portes NOR en série.



Supposons que tous les fractionnements sont soit finaux soit uniformément bons. Désignons par $x_o\ x_1 \ldots . x_n$ les variables ayant défini les fractionnements successifs. Supposons enfin que la chaîne est maximale :

(f_o est une sortie accessible, f_{n+1} et g_{n+1} sont des entrées)

Théorème :

Toutes les portes P_i peuvent être testées par variation des entrées de P_n.

Ceci se démontra par récurrence

(a) P_n définit un fractionnement final donc les variations de g_{n+1} peuvent être observées sur f_n (en donnant à f_{n+1} la valeur zéro)

(b) Supposons que f_{i+1} varie quand g_{n+1} varie. Alors f_i varie.
En effet P_i définit

- soit un fractionnement final donc g_{i+1} est une variable indépendante qui peut être mise à zéro
- soit un fractionnement uniformément bon et en particulier tel que $f_{i+1} \cdot \underset{g_{n+1}}{\text{adher}}\,(g_{i+1}) = 0$

donc quand f_{i+1} varie, g_{i+1} ne masque pas ces variations.

Remarque.
Il n'est fait aucune hypothèse sur la nature du réseau, arborescent ou non. Chacune des fonctions g_i ci-dessus peut être élaborée de manière quelconque, indépendamment ou non de la chaîne C.

5-3-7 Exemple.

$$f = xy + x'y'$$

à réaliser au moyen d'opérateurs NOR.

$$f' = xy' + yx'$$

fractionné en $f_1 = xy'$ $\qquad$ $f_2 = x'y$

fractionnement qui est bon pour f_1 et pour f_2

En effet :

$$\underset{x}{\text{var}}(f_1).f_2 = y'.(x'y) = 0$$

$$\underset{y}{\text{var}}\, f_1.f_2 = x.(x'y) = 0$$

et de même, par raison de symétrie

$$\underset{x}{\text{var}}(f_2).f_1 = 0$$

$$\underset{y}{\text{var}}(f_2).f_1 = 0$$

Itérons le fractionnement sur f_1 :

$$f'_1 = x' + y = x'y' + y$$

que nous fractionnons en

$$f_{11} = x'y' \qquad f_{12} = y$$

Ce fractionnement est final.

De même, fractionnons

$$f'_2 = x'y' + x$$

$$f_{21} = x'y' = f_{11} \qquad f_{22} = x$$

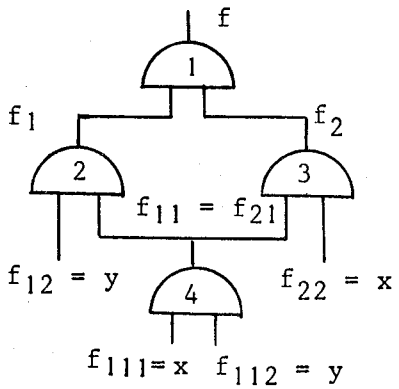
et ce fractionnement est encore final.

Enfin f_{11} est fractionné en

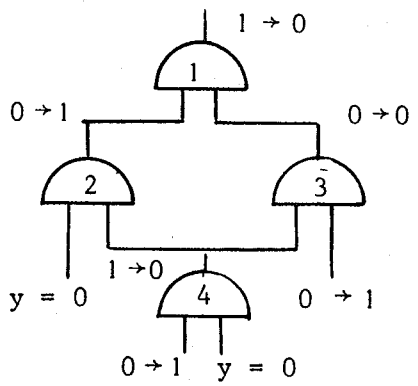
$$f_{111} = x \qquad f_{112} = y$$

fractionnement qui est final.

Le schéma obtenu est donc le suivant :



On peut tester la chaîne des portes 1-2-4 en faisant varier x, à condition de donner à toutes les entrées apparaissant dans les fractionnements finaux, c'est-à-dire ici à y la valeur zéro.



On constate que tout le long de la chaîne 1-2-4 la variation de x entraîne une variation de la sortie de chaque porte. Par contre, les variations de x intervenant sur la chaîne 1-3-4 ne masquent pas les variations précédentes.

5 - 4 Généralisation.

Ce qui précède doit être généralisé en ce qui concerne les opérateurs et en ce qui concerne les fonctions.

5-4-1 Opérateurs quelconques.

(1) Nous avons étudié le cas d'une porte "OU" à deux entrées. Ceci peut être généralisé à une porte OU à un nombre quelconque d'entrées. Définissons le fractionnement de f en $f_1 + f_2 + \ldots\ldots + f_k$ de la manière suivante : on fractionne au préalable f en $f_1 + g_1$ avec la condition $\text{Adher}(f_1).g_1 = 0$. On fractionne ensuite g_1 en $f_2 + g_2$ en respectant $\text{Adher}(f_2).g_2 = 0$, et ainsi de suite.

On constate alors que si on a :

$$\text{Adher}(f_1) \cdot g_1 = 0$$

on aura a fortiori

$$\text{Adher}(f_1) \cdot f_2 = 0$$
$$\text{Adher}(f_1) \cdot f_3 = 0 \quad \text{etc.....}$$

Puisque $f_2 \cdot f_3 \ldots\ldots$ sont inférieurs à g_1, autrement dit f_2, $f_3 \ldots$ ne masquent pas les variations de f_1.

On démontre de même que les variations de f_2, $f_3 \ldots\ldots$ ne sont pas masquées et donc le fractionnement ainsi défini est bon pour tous les f_i.

(2) Ce qui précède peut être généralisé évidemment aux opérateurs ET, NAND, NOR par simple dualisation ou complémentation.

(3) Ce qui précède peut enfin être appliqué aux opérateurs privilégiés quelconques : la décomposition par un opérateur privilégié pouvant toujours se ramener à une suite de décompositions suivant des opérateurs "ET" et "OU".

5-4-2 Fonctions incomplètes.

Nous avons étudié uniquement le cas de fonctions complètes. En fait si on cherche à fractionner une fonction complète $F = \{\underline{f}, \overline{f}\}$ en somme par exemple, il suffira de fractionner $\overline{f}$ en respectant les conditions précédentes :

soit $\underline{f}$ est fractionné en $\underline{f_1} + \underline{f_2}$

$\overline{f}$ " " en $\overline{f_1} + \overline{f_2}$

avec $\text{Adher}(\overline{f_1}) \cdot \overline{f_2} = 0$ alors, a fortiori, pour toutes solutions g_1 et g_3 de F_1 et F_2 on aura

$$\text{Adher}(g_1) \cdot g_2 = 0$$

et le fractionnement sera bon pour g_1 et g_2.

CHAPITRE 6

SYSTEMES ASYNCHRONES - DEFINITIONS

Pour reprendre les notations habituelles, rappellons qu'un système séquentiel est défini mathématiquement par la donnée des 5 éléments suivants :

- X "l'alphabet d'entrée" ou ensemble des valeurs prises par les variables d'entrées $x_1 \ldots\ldots x_n$ du système
- Y l'ensemble des états internes
- Z "l'alphabet de sortie" ou ensemble des valeurs prises par les sorties $z_1 \ldots\ldots z_p$ du système
- F une application de $E = X \times Y$ sur Z
- G une application de E dans Y

Parfois F et G ne sont pas définis sur tout E. En particulier dans le cas des systèmes asynchrones, on admet souvent qu'une seule variable x_i de X peut varier à la fois, ce qui limite considérablement le domaine de définition de G

6 - 1 Représentations

1) Nous utilisons la représentation classique en tableau d'états qui permet aisément une "vue d'ensemble" de l'automate en matérialisant par ses lignes et ses colonnes les caractéristiques essentielles du système.

Pour définir ce tableau T on opère comme suit :

- on fait choix d'un ordre de X_i de X et des Y_j de Y
- une ligne L_i de T associée à l'état interne Y_i de A sera la liste ordonnée des successeurs $G(Y_i, X_j)$ pour tous les X_j successifs
- une colonne C_j de T associée à la valeur X_j des entrées sera la liste des $G(Y_i, X_j)$ pour les Y_i successifs.

2) La table définissant les sorties sera définie de la même manière en remplaçant G par F dans la définition ci-dessus

6 - 2 Définitions diverses.

1) Etat total.

On appelle "état total" ou plus simplement "état" le couple E formé d'un état interne Y_i et d'une valeur X_j d'entrée.

2) Etat stable.

Un état total $E_{ij} = \{Y_i, X_j\}$ sera dit "stable" si on a la relation suivante :

$$G(Y_i, X_j) = Y_i$$

3) Etat instable, transition.

Un état $E_{ij} = \{Y_i, X_j\}$ sera instable si $G(Y_i, X_j) \neq Y_i$

Un tel état sera en général assimilé à la transition de l'état interne Y_i vers l'état interne $Y_k = G(Y_i, X_j)$.

4) Etat indifférent.

$E_{ij} = \{Y_i, X_j\}$ sera indifférent si $G(Y_i, X_j)$ n'est pas défini. Un état interne pourra être physiquement indifférent soit parce que la valeur X_j ne peut pas être prise quand A est dans l'état Y_i (on dira

que l'état total E_{ij} est "inaccessible"), soit parce qu'effectivement on ne se soucie pas de l'évolution de l'automate à partir de l'état interne E_{ij}.

5) Etat interne primitif.

Y_i sera dit primitif s'il existe un et un seul X_j tel que $\{Y_i, X_j\}$ soit stable. Une représentation sera dite primitive si tous ses états internes sont primitifs.

6 - 3 Relations.

Une relation R sur l'ensemble $Y = \{Y_1 \ldots\ldots Y_n\}$ est un ensemble de couples ordonnés d'éléments de Y. Si Y_i, Y_j est un de ces couple, on notera

$$Y_i \quad R \quad Y_j$$

Définition (1) : relation itérante.

On appelle relation itérante sur un automate A, une relation entre états internes vérifiant :

$$(1)\ Y_i \quad R \quad Y_j \Longleftrightarrow \forall X_k,\ G(Y_i, X_k)\ R\ G(Y_j, X_k)$$

Définition (2) : relation précisée.

Une relation R itérante sera précisée par une relation R1 si :

$$Y_i \quad R \quad Y_j \Longleftrightarrow \forall X_k,\ G(Y_i, X_k)\ R\ G(Y_j, X_k)$$

et

$$\forall X_k,\ F(Y_i, X_k)\ R1\ F(Y_j, X_k)$$

Exemple : on définit habituellement la relation d'équivalence entre états internes par

$$Y_i \sim Y_j \Leftrightarrow \forall X_k \quad \begin{array}{l} G(Y_i, X_k) \sim G(Y_j, X_k) \\ \text{et} \\ F(Y_i, X_k) = F(Y_i, X_k) \end{array}$$

La relation d'équivalence est précisée par l'égalité des sorties.

Relation entre états totaux.

Si R1 est une relation entre états internes et R2 une relation entre entrées, on définit une relation R = R1.R2 entre états totaux par

$$\{Y_i, X_k\} \; R \; \{Y_j, X_l\} \iff$$

$$Y_i \quad R1 \quad Y_j \quad \text{et} \quad X_k \quad R2 \; X1$$

Exemple.

On trouve dans la théorie de R.DAVID la notion d'hypovalence. Cette relation est définie à partir des séquences d'entrées et sorties.

$\{Y_i, X_k\} \lhd \{Y_j, X_k\}$ si toute séquence d'entrée applicable à Y_i et à Y_j donne des séquences de sorties, S_i et S_j vérifiant :

$$S_i \leq S_j$$

Cette définition est équivalente à la suivante :

$\{Y_i, X_k\} \lhd \{Y_j, X\}$ si et seulement si :

a) $X_k = X_l$: les deux états sont dans la même colonne

b) $\forall X_m$, $G(Y_i, X_m) \lhd G(Y_j, X_m)$: la relation est itérante.

c) $F(Y_i) \leq F(Y_j)$: les sorties sont comparables.

Relation précisée normale.

Une relation précisée sera dite normale si la relation précisante R1 sur les sorties a la propriété suivante :

si $S = \{s_1 \; s_2 \ldots\ldots s_q\}$

$S_1 \quad R1 \quad S_2 \Leftrightarrow \forall_i, \; s_{1i} \quad r \quad s_{2i}$

r étant la même relation pour toutes les sorties élémentaires.

Ø-relation.

On est fréquemment ammené à étudier des ensembles de valeurs dont certaines ne sont pas définies (on note Ø une telle valeur).

Sur un tel ensemble une relation binaire R sera appelée une Ø-relation si elle est toujours vraie quand un des éléments est non défini.

En particulier, sur un automate on rencontrera deux Ø-relations particulières entre états internes ou totaux, extension des relations d'ordre et d'équivalence.

a) On note $\underset{\sim}{\sim}$ la relation définie par

$A \underset{\sim}{\sim} B \Leftrightarrow A = B$ ou $A = \emptyset$ ou $B = \emptyset$

b) On note $\overset{<}{\sim}$ la relation définie par

$A \lesssim B \Leftrightarrow A \leq B$ ou $A \neq \emptyset$ ou $B = \emptyset$

Demie $\emptyset$ relation.

Une relation R sera appelée une demie $\emptyset$ relation si on a toujours

$$a \; R \; \emptyset$$

(mais pas forcément $\emptyset$ R a)

6 - 4 Relations d'hypovalence, application.

Le point de départ de la méthode présentée ici est la théorie de R.DAVID présentée essentiellement dans sa thèse. DAVID utilise une relation d'ordre entre états totaux qu'il nomme hypovalence. Nous avons envisagé une généralisation de cette méthode, d'une part pour les besoins du calcul automatique, d'autre part pour pouvoir aborder des algorithmes plus élaborés de synthèse.

a) Besoins du calcul automatique.

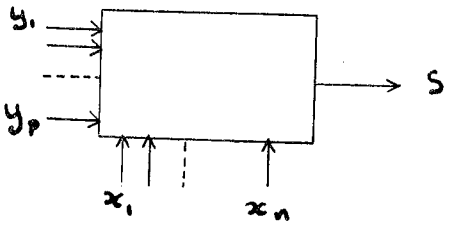
Cherchant à programmer la méthode de DAVID telle qu'elle était à l'origine, il nous est vite apparu que certaines améliorations pouvaient être apportées : la méthode initiale est faite pour être pratiquée "à la main" : l'homme peut avoir une vue d'ensemble sur un tableau d'état que n'a pas l'ordinateur. Si, par exemple, on désire rechercher toutes les relations d'hypovalence au sens de DAVID, on est amené à comparer des états totaux stables d'une même colonne. Cette dernière condition d'être dans une même colonne n'est pas élémentaire pour l'ordinateur ; au moment d'écrire l'instruction de vérification de cette condition, on est évidemment conduit à repenser son utilité, et à chercher à la supprimer.

b) Dans la méthode initiale interviennent deux types de simplifications, et donc à l'origine deux algorithmes distincts, l'un dit de

recherche des états sources, l'autre dit de groupement d'états. Ces deux algorithmes distincts ont pu être présentés comme une application d'une même théorie, comme nous allons le voir.

6-4-1 Cellule séquentielle de DAVID.

Définissons logiquement un opérateur séquentiel de la manière suivante :



C'est un élément de mémoire à n + p entrées, une sortie et deux états.

Les entrées se séparent en deux classes :

- les entrées dites primaires $x_1 \ldots\ldots x_n$ dont l'ensemble des valeurs possibles forment l'ensemble X ;
- les entrées dites secondaires $y_1 \ldots\ldots y_p$

L'état y_o du système peut prendre deux valeurs 0 et 1. Il est défini de la manière suivante :

$$y_o = (y_o + \sum_{i=1}^{p} y_i) \cdot \prod_{j=1}^{n} x'j$$

La sortie est égale à y_o.

Cet opérateur est appelé 'CUSA' (Cellule Universelle pour Séquences Asynchrones)

6-4-2 Utilisation de la cellule.

1) Définition : état simple.

On dira qu'un état Y_i est simple si l'ensemble des X_k tels que

$$G(Y_i, X_k) = Y_i$$

est caractérisé par un monôme unique M_i et si la sortie est indépendante de X_k.

Exemples : un état primitif est simple.

Le groupement de deux états primitifs voisins est un état simple.

2) Soit A un automate représenté par un ensemble d'états simples.

a) A tout état simple Y_i de A on associe une cellule C_i dont les entrées primaires seront les compléments des variables présentes dans M_i. La sortie de C_i est une "variable interne" désignée par Y_i.

b) A toute transition de Y_i vers Y_j on associe une connection entre la sortie de Y_i et l'entrée secondaire de Y_j.

Toute sortie s_p est de la forme :

$$s_p = \sum_q s_p(Y_q) \cdot y_q$$

Méthode de simplification.

On part d'une réalisation dite canonique, où tout état interne est primitif et on cherche des simplifications du réseau ainsi obtenu.

Ces simplifications sont de quatre types :

a) suppression de connections aux entrées primaires

b) suppressions de connections aux entrées secondaires

c) suppression de cellules par groupement d'états

d) suppression de cellules inutiles

Définition.

On appelle hypovalence généralisée une $\emptyset$ relation itérante entre états internes d'un automate précisée par la relation d'ordre habituel sur les sorties :

$$Y_i \lhd Y_j \iff \text{a) } \forall X_k,\ F(Y_i, X_k) \leq F(Y_j, X_k)$$

$$\text{b) } \forall X_k,\ G(Y_i, X_k) \lhd G(Y_j, Y_k)$$

$$\text{ou bien } G(Y_i, X_k) = \emptyset$$

$$\text{ou bien } G(Y_j, X_k) = \emptyset$$

Remarque.

Cette hypovalence est une généralisation de celle développée par DAVID, on n'impose plus ici que les états Y_i et Y_j soient stables

dans une même colonne.

Note : par abus de langage l'hypovalence généralisée sera appelée plus simplement "hypovalence".

6-4-3 Théorème fondamental.

Soit A un automate dont deux états internes Y_i et Y_j vérifient

$$Y_i \lhd Y_j$$

Appelons y_i et y_j les variables internes associées aux états Y_i et Y_j.

On peut, sans modifier le comportement externe de l'automate, changer y_i en $y_i^* = y_i + \lambda y_j$

ceci quelle que soit la valeur de la grandeur booléenne λ (fonction éventuellement des états ou des variables d'entrées)

En effet, ce théorème se démontre par récurrence.

(1) Les sorties ne peuvent être modifiées par la transformation de l'équation de y_i. Si s_q est l'une d'elle, on a :

$$s_q = s_q(Y_i) \cdot y_i + s_q(Y_j) y_j + \sum_{p \neq i,j} s_q(Y_p) \cdot y_p$$

Deux cas peuvent alors se présenter.

(a) $s_q(Y_i) = 0$ Alors la modification apportée à y_i ne peut affecter s_q

(b) $s_q(Y_i) = 1$ Mais alors $s_q(Y_j) = 1$ puisque $Y_i \lhd Y_j$ et on a donc l'expression :

$$s_q = y_i + y_j + \sum_{p \neq i,j} s_q(Y_p) y_p$$
$$= y_i + (y_i + \lambda y_j) + \sum_{p \neq i,j} s_q(Y_p)\, y_p$$

(2) Remarque.

Dire que y_i est remplacé par $y_i^* = y_i + \lambda y_j$ équivaut à dire que quand l'automate est dans l'état y_j, et pour au moins une valeur X_o des entrées X, non seulement $y_j = 1$, mais encore $y_i = 1$

Supposons alors à un instant donné, l'automate dans l'état Y_j, les entrées X ayant la valeur X_o, et étudions l'effet de la transition $X = X_o \longrightarrow X = X_1$.

Appelons respectivement Y_k et Y_ℓ les descendants de Y_i et Y_j pour la valeur X_1.

Les expressions de y_k et y_e sont de la forme

$$y_k = (y_k + y_i + \sum_p y_p) M_k$$
$$y_\ell = (y_\ell + y_j + \sum_q y_q) M_\ell$$

Autrement dit, si y_i est remplacé par y_i^*, quand l'automate passe de l'état Y_j à l'état Y_ℓ à cause de la variation $X_o \longrightarrow X_1$ de X, non seulement y_ℓ va prendre la valeur 1, mais également y_k prendra cette valeur.

D'après la remarque ci-dessus, cela signifie que y_k est changé en

$$y_k^* = y_k + \mu y$$

μ est ici fonction en principe de X (puisqu'en général Y_k ne sera pas descendant de Y_i pour toutes les valeurs de X), et de plus il est fonction des séquences appliquées : μ n'est égal à 1 que si on arrive en Y_ℓ en provenance de Y_j.

Mais puisque $Y_i \lhd Y_j$ leurs descendants Y_k et Y_ℓ vérifient $Y_k \lhd Y$.

On est donc dans les conditions d'une itération du théorème.

6-4-4 Corollaire 1 : états sources.

Définition.

On appelle état minimum, un état Y_i hypovalent à tout état Y_j tel que $\exists X_k$, avec $G(Y_i, X_k) = Y_i$ et $G(Y_j, X_k) = Y_j$ (Y_i et Y_j sont stables dans une même colonne)

Théorème de DAVID.

> Si un état Y_i est minimum, on peut supprimer toutes les entrées secondaires pour les remplacer par une fonction unique identique à 1. Un tel état est appelé état source.

En effet :

Lemme.

Si Y_i est un état minimum et Y_p un état quelconque, on peut relier une entrée secondaire de Y_i à la sortie de Y_p ; trois cas peuvent en effet se produire :

a) $\exists X_k$, $G(Y_p, X_k) = Y_i$

Alors la connexion existe déjà et la propriété est banale.

b) $\forall X_k$ tel que $G(Y_i, X_k) = Y_i$, on a :

$$G(Y_p, X_k) = Y_j \neq Y_i$$

Mais alors pour cette même valeur X_k, Y_j est stable :

$G(Y_j, X_k) = Y_j$ et, d'après la définition de Y_i, on a :

$$Y_i \lhd Y_j$$

Alors la propriété devient un corollaire immédiat du théorème de base.

c) $\forall X_k$ avec $G(Y_i, X_k) = Y_i$ on a :

$$G(Y_p, X_k) = \emptyset$$

Puisqu'il n'y a pas spécification, rien n'empêche le descendant de Y_p d'être justement Y_i pour la valeur X_k.

Corollaire.

L'équation de la variable y_i peut donc s'écrire :

$$y_i = (\sum_{j \in J} y_j) M_i$$

Le signe Σ étant étendu à l'ensemble de tous les états.

Mais $\Sigma y_j = 1$, donc les entrées secondaires peuvent être toutes supprimées et remplacées par 1, d'où le théorème.

Application.

On en déduit une première méthode de simplification par suppression de connexions vers les entrées secondaires, méthode analogue à celle présentée par R.DAVID.

Exemple : soit l'automate ci-dessous.

xy	00	01	11	10	$S_1 S_2 S_3$
	(a)	(a)	e	e	0 0 0
	d	(b)	(b)	-	1 1 0
	(c)	(c)	e	f	0 1 0
	(d)	b	-	d	1 1 1
	c	a	(e)	(e)	1 0 0
	c		b	(f)	1 0 1

Montrons que a est un état source:

a doit être comparé aux états qui sont stables dans une même colonne que a c'est-à-dire à b, c, d

$a \lhd b \Rightarrow a \lhd d,\ e \lhd b$

$a \lhd d \Rightarrow a \lhd b,\ e \lhd d$

$e \lhd b \Rightarrow a \lhd b,\ c \lhd d$

$e \lhd d \Rightarrow c \lhd d,\ a \lhd b$

$c \lhd d \Rightarrow c \lhd b$

$c \lhd b \Rightarrow c \lhd d,\ e \lhd b$

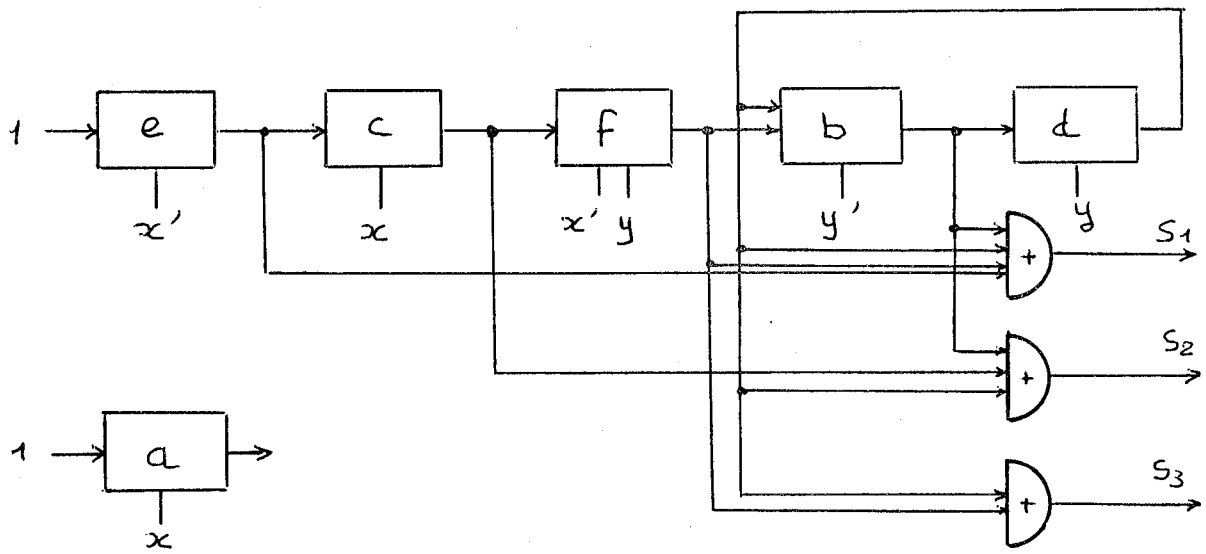
$a \lhd c \Rightarrow e \lhd f$

$e \lhd f \Rightarrow e \lhd b$

Autrement dit a est bien un état minimum, donc source.

Par la même relation on montre que e est également minimum et donc source.

L'automate peut être présenté par :



On constate que la cellule associée à l'état a est inutile, n'étant reliée à rien.

Remarque.

Dans notre théorie, telle qu'elle est utilisée ici, nous pouvons traiter un exemple d'automate qui au départ, n'est pas primitif. Dans la méthode initiale de DAVID on aurait du le rendre primitif en dédoublant les états a, b, c, d, e et nous aurions obtenu un automate à 11 états plus lourd à traiter.

6-4-5 Corollaire 2 : Suppression d'entrées primaires

Théorème.

Soit A un automate, Y_i et Y_j deux états de A vérifiant :

a) $Y_i \lhd Y_j$

b) Y_i est stable sur l'ensemble des X_k caractérisé par le monôme M_i

c) $\forall X_k$ avec $M_j(X_k) = 1$ on a :
$G(Y_i, X_k) = Y_j$ ou $\emptyset$

d) $\nexists Y_p$, X_k et X_l tels que
$G(Y_p, X_k) = Y_p$
$M_j(X_k) = 1$
$G(Y_p, X_l) = Y_i$

Alors on peut changer dans l'expression de y_i, M_i en $M_i + M_j$ sans rien changer au fonctionnement externe de l'automate.

En effet, y_i peut s'écrire :

$$y_i = (y_i + y_p) M_i$$

Développons l'expression des y_p :

$$\forall p,\ y_p = k_p \cdot M_p$$

Donc $y_i = (y_i + \Sigma k_p M_p) M_i$

Changeons M_i en $M_i + M_j$. On obtient :

$$y_i^* = (y_i + \Sigma k_p M_p)(M_i + M_j)$$

$$= (y_i + \Sigma y_p) M_i + y_i M_j + \Sigma k_p M_p M_j$$

Or, d'après la condition (d), le produit $M_p.M_j$ est nul. Donc :

$$y_i^{\star} = (y_i + \Sigma y_p)M_i + y_i.M_j$$

Or $y_i.M_j$ est inférieur à y_j d'après (c) :

$$y_j = (y_j + y_i + \Sigma y_q)M_j$$

On peut donc écrire :

$$y_i M_j = \lambda y_i$$

Et la modification apportée change

$$y_i \text{ en } y_i^{\star} = y_i + \lambda y_j$$

D'après la condition (a) et le théorème de base, ceci ne change pas les séquences de sortie de l'automate.

Remarque.

La démonstration ne fait pas intervenir le fait que M_i et M_j sont des monômes : cette condition est en fait imposée uniquement par la technologie de la cellule séquentielle.

En pratique, pour que la modification soit intéressante, on ajoutera la condition (e)

e) $M_i + M_j \geq M_k > M_i$, M_k étant seul obligatoirement un monôme.

Par exemple :

$$M_i = Mx \quad M_j = Mx'$$

$$M_i + M_j = M$$

Notation.

L'ensemble des conditions a,b,c,d,e seront notées

$$Y_i \lhd Y_j$$

Exemple d'application.

On trouve dans la thèse de R.DAVID un exemple réel d'automate dans lequel on trouve un état interne noté a, stable sur 2^p-1 points

caractérisés par une fonction f qui est un monôme M moins un point X_o, c'est-à-dire :

1) $\forall X \neq X_o$ et $M(X) = 1$, on a :

$$G(a, X) = a$$

2) $G(a, X_o) = f$

La fonction caractéristique de l'ensemble des X en a est stable, n'étant pas un monôme unique, a n'est pas un état simple et ne peut donc, en principe pas être représenté par une cellule unique.

Or, dans cet exemple, a est hypovalent à f. On peut donc remplacer f_i par $f_i^{\times} = f_i + m_o$.

m_o étant le monôme caractéristique de X_o, mais $f_i + m_o = M$, a devient donc maintenant un état simple qui peut être représenté par une cellule unique.

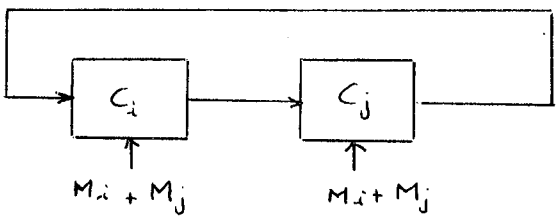
6-4-6 Corollaire 3. Groupements d'états.

La propriété précédente permet de trouver la simplification par groupement d'états.

Soient Y_i et Y_j deux états simples, M_i et M_j leurs monômes caractéristiques et supposons

$$Y_i \not\lhd Y_j \text{ et } Y_j \not\lhd Y_i$$

On peut remplacer M_i et M_j par $M_i + M_j$



On constate que, dès que $y_i = 1$, y_j devient égal à 1 et inversement. On peut donc remplacer l'ensemble des deux cellules C_i et C_j par une seule que nous appellerons C_{ij}, d'entrée $M_i + M_j$. On retrouve la propriété énoncée par DAVID comme conséquence du théorème de base.

EXEMPLE:
AUTOMATE ASYNCHRONE AYANT:
3 ENTREES PRIMAIRES,
2 SORTIES,
13 ETATS INTERNES,
L'AUTOMATE EST DONNE PAR UNE TABLE D'ETATS PRIMITIVE:

000	001	011	010	110	111	101	100	S2,S1
11				4		2	(1)	01
	6				3	(2)	1	01
		7		4	(3)	2	1	01
			9	(4)	3		1	01
(5)	6		13				1	01
5	(6)	7				2		11
	6	(7)	9		3			11
		(8)	9		3			01
11		10	(9)	4				10
	12	(10)	13		3			11
(11)	12		13				1	00
11	(12)	8				2		10
11			(13)	4				00

SORTIE DE :	EST CONNECTEE A L'ENTREE DE:
1	9 6 6 S2
5	6 S2
6	9 5 S1 S2
8	9 S2
9	10 S1
10	S1 S2
12	8 S1

C H A P I T R E 7

THEORIE DU CONSENSUS DANS UN AUTOMATE

7 - 1 Introduction à l'idée de consensus.

7-1-1 Technique de groupement.

Dans la théorie précédente, nous avons vu apparaître une opération de groupement d'états qui s'apparente beaucoup au groupement de monômes de l'Algèbre de Boole.

Si deux états Y_i et Y_j sont stables pour des valeurs de X caractérisées respectivement par les monômes $M_i = MX$ et $M_j = MX'$, on peut envisager (si les conditions sur les sorties, les descendants et les ascendants sont vérifiées) de grouper Y_i et Y_j en un seul état Y_{ij} caractérisé par le monôme :

$$M_{ij} = M_i + M_j = MX + MX' = M$$

Tout comme en algèbre de Boole, on peut alors faire la remarque suivante : la méthode de groupement d'états donnera les états maximaux à condition de partir d'un automate primitif (sous une forme équivalente à la forme canonique de l'algèbre de Boole). Par contre, si on part d'un automate déjà réduit et mal réduit, on a peu de chances d'arriver aux états maximaux et donc à la forme la plus économique.

Exemple :

00	01	11	10	S
	(a)	b		0
	a	(b)	c	0
d		b	(c)	0
(d)	e		c	1
	(e)			1

se réduit à

00	01	11	10	S
	(ab)	(ab)	bc	0
de	ab	(bc)	(bc)	0
(de)	(de)		bc	1

Par contre, si on part de la forme déjà réduite ci-dessous, on ne peut arriver au même résultat sans dédoubler au préalable les états bc

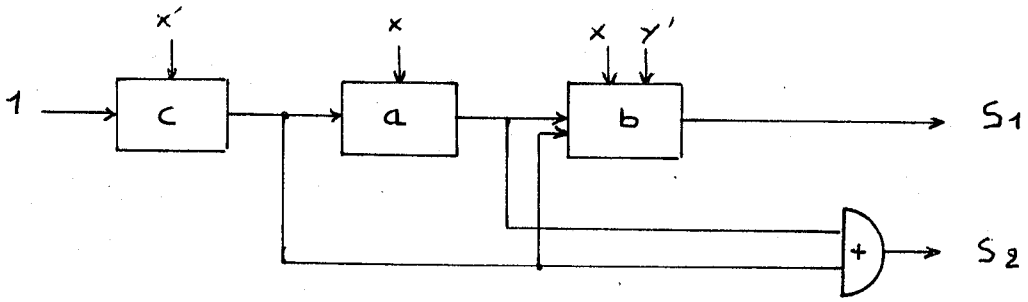
00	01	11	10	S
	(a)	b		0
c	a	(bc)	(bc)	0
(de)	(de)			1

7-1-2 Cablage entre cellules et codage.

Dans la méthode telle qu'elle est présentée dans le chapitre 6, le point de départ est un codage canonique des états : à chaque état est associée une cellule, et le réseau de cellules est semblable au graphe des transitions.

Pourtant, utilisant les mêmes cellules que précédemment, et les mêmes règles pratiques de cablage, il est possible de réaliser des réseaux plus généraux :

Exemple :



Ce réseau réalise l'automate ci-dessous :

00	01	11	10	S_1S_2
(Y_1)	Y_2		Y_4	01
Y_1	(Y_2)	Y_4		11
	(Y_3)	Y_4		10
Y_1	Y_2	(Y_4)	(Y_4)	01

Dans cet exemple Y_1 est codé par $y_a = 1$

Y_2 " " " y_a et $y_b = 1$

Y_3 " " " $y_b = 1$

Y_4 " " " $y_c = 1$

et la réalisation ci-dessus ne peut être obtenue par la méthode de DAVID.

En fait ici Y_2 doit être considéré comme la "somme" des états Y_1 et Y_3. Une nouvelle théorie, celle du consensus entre états va permettre des réductions nouvelles.

7-1-3 Consensus de Pichat.

Nous utiliserons une théorie développée par E.PICHAT qui généralise la notion de consensus, pour permettre la démonstration dans un algorithme très efficace de recherche d'éléments maximaux.

Rappelons brièvement le principe de la généralisation : PICHAT étend la notion de monôme en définissant des "pavés" qui sont des n - uples $\{a_o\ a_1 \ldots\ldots a_n\}$ dont chaque a_i appartient à un treillis distributif (et non forcément au treillis de Boole de la forme {0, x, x', 1}). Les relations d'ordre de chaque treillis induisent une relation d'ordre entre pavés, et PICHAT démontre un algorithme appelé "algorithme de sélection" qui permet d'obtenir les pavés maximaux.

Définitions.

1) Nous appellerons variable interne la grandeur booléenne y_i présente à la sortie d'une cellule C_i.

2) On dira que y_a est "ascendant" de y_b s'il existe une connexion allant de la sortie de C_a à une entrée secondaire de C_b.

3) On dira que y_b est descendant de y_a pour la valeur X_k des entrées X si :

a) y_b est descendant de y_a

b) quand $y_a = 1$, y_b prend la valeur 1 dès que X prend la valeur X_k.

En général plusieurs variables $y_{b1}.....y_{bp}$ peuvent être des descendantes de y_a pour la valeur X_k.

On notera :

$$G(Y_a, X_k) = \{y_{b1}, y_{b2}, ... y_{bp}\}$$

(G est une application de l'ensemble produit $\{y_i\} \times \{X_j\}$ dans l'ensemble $P(\{y_i\})$ des parties de $\{y_i\}$). Nous noterons $G(y_a)$ l'ensemble de tous les descendants : $G(y_a) = \bigcup_k G(y_a, X_k)$.

4) Une variable y_i sera dite stable pour la valeur X_k des entrées si, à partir de la valeur $y_i = 1$ et quand X prend la valeur X_k, y_i ne change pas. On conviendra de dire que y_i est son propre descendant pour X_k, ce qui n'est pas en contradiction avec 2) puisqu'une connexion existe (à l'intérieur de la cellule) entre la sortie C_i et une entrée secondaire.

Nous caractérisons l'ensemble des X_k pour lesquels y_i est stable par une fonction booléenne $S(y_i, X)$ telle que :

$$S(y_i, X_k) = 1 \iff y_i \text{ est stable pour } X_k$$

En pratique les cellules sont construites de telle manière que la fonction S soit obligatoirement un monôme en X :

$$\forall y_i,\ S(y_i, X) = M_i(X)$$

5) Descendants nuls descendants non spécifiés.

Il est possible qu'une variable y_a n'ait aucun descendant pour une valeur X_k. On exprimera ceci par :

$$G(y_a, X_k) = "0"$$

On prendra soin de distinguer cette notion d'absence de descendant de la notion de descendant non spécifié, notée :

$$G(y_a, X_k) = "\phi"$$

qui signifie qu'il y a un choix possible.

Enfin on distinguera cette valeur "Ø" de la valeur "1" indiquant l'ensemble de toutes les variables. Si $G(y_i, X_k)$ = "1" toutes les variables sont des descendants obligés de y_i, si $G(y_i, X_k)$ = "Ø" on peut choisir parmi les variables des descendants quelconques.

Pour des raisons pratiques, on considère que la valeur "Ø" est meilleure que la valeur "1" et on note "Ø" > "1".

En pratique le symbole Ø sera souvent remplacé par un blanc dans l'écriture des variables.

6) Sorties.

Nous nous efforçons d'obtenir des réalisations dans lesquelles chaque sortie est une somme de variables internes. Pour chaque sortie élémentaire s_i, nous aurons :

$$s_i = \Sigma\, r_{ij}\, y_j$$

Chaque s_i est donc défini par l'ensemble des r_{ij} qui sera appelé "pseudo sortie".

Pour un indice i donné, l'ensemble $\{r_{ij}\}$ est une fonction de j notée :

$$r_i = f_i(j)$$

L'ensemble des r_i est également une fonction de j notée :

$$R = F(j)$$

La valeur de R pour un j donné sera appelée la "sortie" de y_j.

7) Table des variables.

Nous utiliserons une représentation analogue au tableau d'état pour représenter les interconnexions entre variables ou entre variables et sorties. On définit une "table des variables" comme un tableau rectangulaire T dont une ligne est associée à chaque variable :

a) $T(i,j) = G(y_i, X_j)$

(les variables stables étant entourées d'un cercle)

b) $T(i, n+1) = F(i)$

Exemple : l'automate du paragraphe 7-1-2 peut être représenté par la table des variables ci-dessous :

00	01	11	10	$r_1 r_2$
(y_a)	$(y_a)\, y_b$	y_c		01
	(y_b)	y_c		10
y_a	y_b	(y_c)	(y_c)	01

Cette table fait ressortir le fait qu'on peut avoir simultanément $y_a = 1$, $y_b = 1$ (ce qui est le code de l'état Y_2).

Remarques.

a) Dans un réseau réalisé par la méthode de DAVID, et avant toute réduction, il y a isomorphisme entre le tableau d'états et la table des variables.

b) Cette forme de la table des variables conduit souvent à dire (en confondant les notions d'états et de variables) que l'automate est "non déterministe", terme qui est évidemment impropre et prête à confusions.

8) Représentation canonique d'une variable.

Une variable interne y_a sera notée

$$y_a = \{a_o, a_1, a_2 \ldots\ldots\ldots a_n\}$$

avec les définitions suivantes.

a) a_o est l'ensemble $\{G(y_a), F(a)\}$ des descendants de la variable y_a et de sa sortie, et sera appelé une ligne (a_o est une ligne du tableau des variables défini précédemment).

b) Pour $i \geq 1$, a_i est un élément du treillis de Boole $t_i = \{0, x_i, x'_i, 1\}$, tel que l'on ait : $\prod_{i=1}^{n} a_i = M_a$

7 - 3 Relations entre variables.

7-3-1 Remplacement.

On dira que l'ensemble $Y_1 = \{y_1 \ldots\ldots y_p\}$ remplace l'ensemble $Y_2 = \{y_q \ldots y_r\}$ pour une valeur X_ℓ si pour tout y_s tel que $Y_2 \subset G(y_s, X_\ell)$, on peut remplacer Y_2 par Y_1 dans $G(y_s, X_\ell)$ sans changer les séquences de sorties de l'automate.

Condition suffisante.

Pour que Y_1 remplace Y_2, il suffit que :

a) Quand $y_q = \ldots\ldots = y_r = 1$, le remplacement ne change pas les sorties, c'est-à-dire :

$$\sum_{i=q}^{r} F(i) = \sum_{j=1}^{p} F(j)$$

b) Dans les mêmes conditions, que pour tout X_k la propriété s'itère sur les descendants de Y_1 et Y_2, c'est-à-dire :

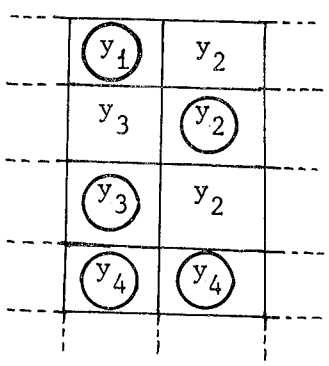
$$\forall X_k, \bigcup_{i=1}^{p} G(y_i, X_k) \text{ remplace } \bigcup_{j=q}^{r} G(y_j, X_k)$$

c) et enfin, qu'on n'introduise pas de fausses transitions vers les $y_1 \ldots y_p$ c'est-à-dire :

$\forall y_i$, $Y_2 \subset G(y_i, X_e)$, $\nexists\, y_j \in Y_1$ tel que y_i et y_j soient stables simultanément.

Cette condition est une généralisation de la condition (d) du paragraphe 6-4-5.

Exemple : dans la configuration suivante



la variable y_4 ne peut remplacer l'ensemble $\{y_2, y_3\}$ même si les descendants et les sorties sont égales. En effet, la cellule de base étant asynchrone, si on remplace la connexion de C_1 à C_2 par une connexion de C_1 à C_4, dès que $y_1 = 1$, y_4 devient égal à 1.

Remarque.

La condition n'est nécessaire que si Y_1 et Y_2 constituent chacun un code d'état. Dans le cas général, si par exemple $y_1 \ldots . y_p$ peuvent ne pas être seuls à 1, certaines inégalités entre sorties pourront être masquées.

7-3-2 Pseudo-équivalence.

On dira que Y_i est pseudo-équivalent pour la valeur X_ℓ à Y_j si Y_i remplace Y_j et Y_j remplace Y_i pour X_ℓ.

7-3-3 Equivalence.

On dira que Y_i est équivalent à Y_j si Y_i est pseudo- équivalent à Y_j pour tout X.

7-3-4 Qualité des variables.

Pour coder les états d'un automate, il est en général possible de choisir un certain nombre de variables internes parmi un grand nombre. Nous cherchons toutefois une réalisation économique et donc nous cherchons un codage à l'aide d'un petit nombre de variables aussi économiques que possible.

Nous dirons que y_a est meilleure que y_b (et on notera ceci $y_a \geqq y_b$) si

a) $F(a) \geqq F(b)$: la variable y_a sert plus souvent, pour l'élaboration des sorties, que y_a .

b) Pour tout E_k, $G(y_b, E_k) \subset G(g_a, E_k)$: y_a est plus souvent utilisé que y_b comme entrée secondaire de cellule.

c) y_a est plus stable que y_b : $M_a \geqq M_b$ (il y a moins de connexions aux entrées primaires de C_a que de C_b).

Avec cette définition, le problème de la minimisation d'un automate se résume aux deux opérations suivantes :

a) recherche de toutes les variables de qualité maximales

b) choix parmi ces variables d'un sous-ensemble minimum permettant de coder l'automate à réaliser.

7 - 4 Treillis des lignes.

1) La relation de qualité entre variables peut se décomposer en un produit de relations d'ordre :

Si $y_a \geq y_b$ c'est-à-dire :

$\{a_o\ a_1 \ldots\ldots\ a_n\} \geq \{b_o, b_1 \ldots\ldots b_n\}$

nous avons par définition de la qualité :

a) $\forall_i$, $1 \leq i \leq n$, $a_i \leq b_i$

puisque $\prod a_i = M_a \leq \prod b_i = M_b$

b) Entre $a_o = \{G(y_a), F(a)\}$ et $b_o = \{G(y_o), F_b\}$ on a une relation définie par :

$$G(y_b) \subset G(y_a)$$
$$F(y_b) \leq F(a)$$

Cette relation entre lignes est comme la qualité d'une relation d'ordre que nous noterons du signe $\leq$.

2) Elément minorant.

L'ensemble des lignes possède un élément minorant "0" qui est la ligne associée à la variable sans descendants, de sortie nulle et stable nulle part.

Cette ligne est évidemment sans intérêt et la variable correspondante ne sort jamais dans un codage.

3) Elément majorant.

L'ensemble des lignes possède un élément majorant "ϕ" associé à la variable qui a des descendants partout non spécifiés, une sortie identique à 1 et qui est stable partout.

Comme précédemment cette variable ne sert jamais sauf pour représenter les connexions non spécifiées.

4) Théorème.

L'ensemble des lignes forme un treillis distributif.

Cela résulte des paragraphes 1, 2 et 3 ci-dessus. Ce treillis sera noté t_o.

5) Somme et produit.

a) On appelle somme de deux lignes $L_a = \{G(y_a), F_{(a)}\}$ et $L_b = \{G(y_b), F_{(b)}\}$ une ligne $L_c = \{G(y_c, F_c)\}$ qui est la borne supérieure de L_a et L_b et qui est donc définie par :

$$G(y_c) = G(y_a) \cup G(y_b)$$

$$F(c) = F(a) + F_b$$

C'est la plus petite ligne supérieure à la fois à L_a et L_b. On notera :

$$L_c = L_a + L_b$$

Remarque.

Dans l'ensemble des lignes associées aux variables codant un automate, on ne trouve pas forcément la ligne somme de deux autres, mais on peut toujours construire une variable nouvelle adjointe au codage et ayant pour ligne la ligne cherchée. (On peut toujours adjoindre à un codage une variable quelconque, quitte, physiquement, à ne pas s'en servir).

b) Produit.

On appelle produit de $L_a = \{G(y_a), F_{(a)}\}$ et $L_b = \{G(y_b), F_{(b)}\}$ leur borne inférieure définie par :

$$L_d = \{G(y_d), F(d)\}$$

$$G(y_d) = G(y_a) \quad G(y_b)$$

$$F(d) = F(a) \cdot F(b)$$

C'est la plus grande ligne inférieure à L_a et à L_b. On notera :

$$L_d = L_a \cdot L_c$$

Comme précédemment y_d n'existe pas toujours, mais peut être rajouté au codage.

7 - 5 Consensus.

1) Remarque.

Chaque variable $y_a = \{a_o\ a_1 \ldots\ldots a_n\}$ est un élément du produit T des treillis t_o, t_1, $\ldots . t_n$. Ce treillis T est muni de la relation $\leq$ produit des relations d'ordre de chaque t_i, qui est donc par construction la relation de qualité du paragraphe 7-3-3.

2) Définition.

On appelle consensus d'ordre i des variables $y_a = \{a_o \ldots\ldots a_n\}$ et $y_b = \{b_o \ldots\ldots b_n\}$ la variable y_c notée :

$$y_c = y_a \underset{i}{*} y_b$$

et définie par :

a) $\forall\ j \neq i,\ c_j = a_j \cdot b_j$

b) $C_i = a_i + b_i$

3) Exemples.

a) Consensus d'ordre i = 0.

Soient les variables définies par les lignes de tableau d'état suivantes :

y_a =	"Ø"	y_d	y_a	y_a	y_b	"Ø"	y_e	"Ø"	01
y_b =	y_g	"Ø"	"Ø"	y_b	y_b	"Ø"	y_f	"Ø"	10

Leur consensus d'ordre Zéro est la variable y_c qui a pour descendant l'union des descendants de y_a et y_b.

On remarque à ce propos que le descendant non spécifié "Φ" étant l'élément majorant du treillis, les descendants de y_c ne sont pas spécifiés dès que ceux de y_a ou de y_b ne sont pas spécifiés.

La sortie de y_c est l'union des sorties de y_a et y_b.

Enfin y_a est stable pour un monôme M_a et y_b pour un monôme M_b, leur consensus est stable pour le monôme $M_a \cdot M_b$ (c'est-à-dire quand y_a et y_b sont stables). La variable y_c est donc :

y_c =	"Ø"	"Ø"	"Ø"	y_a, y_b (y_c)	y_b	"Ø"	y_e, y_f	"Ø"	11

b) Consensus d'ordre i ≠ 0

Soient y_a et y_b définies par

	000	001	011	010	110	111	101	100	$r_1 r_2$
	y_e	(y_a)	(y_a)	y_b	"Ø"	y_e	y_g	"Ø"	01
	y_e	"Ø"	y_a	(y_b)	(y_b)	y_f	Ø	y_h	10

y_a est stable sur le monôme $M_a = x'z$

y_b est stable sur le monôme $M_b = yz'$

Leur consensus d'ordre 3 (par rapport à z) est donc stable pour le monôme $M_c = x'y$. Les descendants de y_c sont l'intersection de ceux de y_a et y_b. La sortie de y_c est le produit des sorties de y_a et y_b.

On a donc :

y_c =	y_e	y_a	y_a (y_c)	y_b (y_c)	y_b	"0"	y_g	y_h	00

Remarque : cet exemple montre que le consensus d'ordre i ≠ 0 n'est pas toujours intéressant en pratique. Nous cherchons à remplacer une variable y_c qui serait consensus de y_a et y_b par l'ensemble $\{y_a, y_b\}$. Or la sortie de y_c est le produit des sorties de y_a et y_b et nous voulons réaliser les sorties avec uniquement des sommes de variables. Le consensus $y_a \overset{*}{_i} y_b$ ne sera donc intéressant que si F(a) = F(b) car alors

F(a) . F(b) = F(a) + F(b)

Règle : on ne fera le consensus d'ordre i ≠ 0 qu'entre variables de sorties égales.

7 - 6 Simplifications.

On voit immédiatement apparaître deux types très élémentaires de simplifications.

7-6-1 Variables instables.

Si, par consensus, on obtient une variable qui n'est stable nulle part, cette variable est sans intérêt pratique et peut être éliminée.

7-6-2 Descendants inutiles.

Si on admet la règle des transitions uniques sur les entrées, certains descendants peuvent être supprimés.

Soit y_a stable pour les X_k tels que $M_a(X_k) = 1$, alors, pour tout X_ℓ tel qu'il n'existe pas un des X_k qui en diffère d'une seule variable, $G(y_a, X_\ell)$ n'a pas à être spécifié.

Exemple.

La variable y_c de l'exemple 2 ci-dessus peut être simplifiée en :

y_e	y_a	$y_a, (y_c)$	$y_b, (y_c)$	y_b	"0"	∅	∅	00

Théorème.

$y_a \underset{o}{*} y_b$ et si de plus $\forall y_j, \forall X_k$, $y_c \in G(y_j, X_k)$ on a :

$X_k)$ et $y_b \in G(y_b, X_k)$ alors $\{y_a, y_b\}$ remplace y_c pour X_k.

des définitions du consensus.

apparaître ici l'intérêt du consensus : si $y_c = y_a \underset{o}{x} y_b$ on

ans certains cas supprimer la cellule C_c.

7-6-4 Théorème.

Si $y_c = y_a \underset{0}{\times} y_b$ alors pour tout X_k tel que y_c est stable, y_c remplace l'ensemble $\{y_a, y_b\}$.

En effet, pour tout X_k où y_c est stable, on a :

$$G(y_c,X_k) = G(y_a,X_k) \cup G(y_b,X_k)$$
$$= \{y_a, y_b, y_c, ...\}$$

et y_a, y_b peuvent être supprimés de cet ensemble puisque :

a) les sorties sont inchangées car

$$F(a) + F(b) + F(c) = F(c)$$

b) les descendants de y_a et y_b sont ceux de y_c

c) il n'est pas introduit de fausse valeur 1 puisqu'aucune connexion n'est rajoutée.

7-6-5 Soient y_a et y_b deux variables ayant (a) des sorties égales et (b) des descendants tels que $\forall X_k$, $G(y_a, X_k) = G(y_b, X_k)$ ou $G(y_a,X_k) = \emptyset$ ou $G(y_b, X_k) = \emptyset$;

Soit $y_c = y_a \underset{i}{\times} y_b$ $\qquad i \neq 0$

1) Sur tout X_k tel que $M_a(X_k) = M_c(X_k) = 1$ on a :

$$G(y_c, X_k) = \{y_c, y_a, ...\}$$

On peut supprimer y_a de cet ensemble; en effet d'après (b), y_c a tous les descendants de y_a et y_b et donc y_c est équivalent à l'ensemble $\{y_a, y_b\}$

2) De la même manière, pour tout X_k tel que $M_b(X_k) = M_c(X_k) = 1$, on peut supprimer y_b de $G(y_c, X_k)$

Exemple.

Dans l'exemple 1 précédent, la variable y_c peut être simplifiée :

y_a =	Ø	y_d	(y_a)	(y_a)	y_b	Ø	y_e	Ø	01
y_b =	y_g	Ø	Ø	(y_b)	(y_b)	Ø	yf	Ø	10
$y_a \overset{*}{\underset{o}{}} y_b$ =	Ø	Ø	Ø	(y_c)	y_b	Ø	y_e,y_f	Ø	11

7-6-6 Théorème.

Si (1) $y_c = y_a \overset{*}{\underset{i}{}} y_b$ $\quad i \neq 0$

(2) y_a et y_b sont pseudo-équivalent, c'est-à-dire

a) $F(a) = F(b)$

b) $\forall X_k$, $G(y_a,X_k) = G(y_b,X_k)$ ou $G(y_a,X_k) =$ "Φ" ou $G(y_b,X_k) =$ "Φ"

(3) $\forall y_i, \forall X_k$, $y_c \in G(y_i, X_k)$ on a: $y_a \in G(y_a, X_k)$ et $y_b \in G(y_b, X_k)$

(Comme précédemment y_a et y_b ne sont pas stables dans une colonne où y_c a un ascendant, alors on peut remplacer toute connexion vers C_c soit par une connexion vers C_a soit par une connexion vers C_b).

En effet, soit y_i tel que $G(y_i, X_k) = y_c$. Deux cas peuvent se présenter

1) $y_a \subset G(y_a, X_k)$ (y_a est stable pour cette valeur).

On peut remplacer toute connexion vers C_c par une connexion vers C_a puisque :

a) cela ne change pas les sorties d'après (2-a)

b) cela ne change pas les descendants. En effet ou bien $\forall X_k$, $G(y_c, X_k) =$ "Ø" et donc la modification n'a pas d'importance, ou bien $\exists X_k$, $G(y_c, X_k) \neq$ "Ø" et alors $G(y_c,X_k) = G(y_a,X_k)$ d'après (2-b)

c) cela n'introduit pas de fausse valeur $y_a = 1$ d'après (3)

2) $y_b \subset G(y_a,X_k)$ et la démonstration est identique

7 - 7 Méthode générale de minimisation.

La méthode se décompose en trois étapes.

7-7-1 Détermination d'un réseau primitif minimum :

a) On utilise la première partie de la méthode de DAVID pour la recherche des hypovalences, puis des états sources. Nous arrivons ainsi à un premier réseau simplifié dans lequel éventuellement de nombreuses connexions ont été supprimées.

b) La table des variables de ce réseau se déduit aisément de la table d'état de l'automate : aux notations près, il suffit de supprimer toutes les transitions vers les variables sources (en prenant garde de les remplacer par "0" et non par ∅).

7-7-2 Algorithme de consensus.

Travaillant maintenant sur les variables internes, on cherche toutes les variables maximales par l'algorithme de sélection par exemple.

Algorithme de sélection.

On trouvera dans la thèse de PICHAT la description précise de l'algorithme de sélection qui est très adapté à notre problème. Rappelons-en brièvement le principe.

On choisit un ordre des éléments de départ (ici les variables internes , en algèbre de Boole, les monômes).

1) On "sélectionne" un premier élément et on envisage son consensus avec tous ceux qui le suivent dans la liste. On effectue au passage toutes les simplifications possibles. Les consensus formés sont rangés à la droite de la liste.

2) On sélectionne le deuxième élément et on fait de même son consensus avec tous ceux à sa droite, et ainsi de suite jusqu'au dernier élément à droite de la liste.

PICHAT démontre que cet algorithme donne tous les éléments maximaux cherchés. L'intérêt de cet algorithme par rapport aux autres est de diminuer le nombre de consensus formés et donc dans notre cas, de diminuer le nombre de comparaisons des nouvelles variables aux anciennes.

7-7-3 On cherche un ensemble minimum de variables de codage, en utilisant à partir des relations de consensus, une méthode classique de couverture des variables initiales :

a) à chaque variable interne y_i on associe une variable booléenne z_i avec la signification suivante :

$z_i = 1$ signifie que y_i est utilisé

$z_i = 0$ signifie que y_i est inutilisé

b) Quand une des variables initiales définie en (1) est consensus de deux variables y_j et y_k et quand elle peut physiquement être remplacée par y_j et y_k on doit utiliser soit y_i, soit y_j et y_k, et donc avoir

$$z_i = 1 \quad \text{ou} \quad z_j \text{ et } z_k = 1$$

ou encore

$$z_i + z_j \cdot z_k = 1$$

Dans le cas général, une variable peut être consensus de plusieurs manières et on aura

$$z_i + \sum_p z_{jp}\, z_{kp} = 1$$

Ceci doit être vrai pour tout y_i et donc on doit avoir :

$$\prod_i (z_i + \sum_p z_{jp} \cdot z_{kp}) = 1$$

En développant cette expression, on obtient un ensemble de monômes dont chacun représente une solution au problème. On pourra envisager une méthode heuristique pour ne chercher que des monômes minimaux sans développer toute l'expression.

7 - 9 Exemple.

Considérons l'automate défini par la table d'état suivante :

000	001	011	010	110	100	101	111	S_1S_2
b	(a)	(a)	f			d	j	01
(b)	a		e		i			00
b			f	(c)	(c)	d	j	01
b	a			c	(d)	(d)	j	01
b		a	(e)	(e)	d		j	11
b	a	(f)	(f)	c			j	01
b		a	(g)	(g)	d		j	10
b	a			g	(h)	(h)	j	10
b				e	(i)			11
						h	(j)	00

Dans cet exemple il n'y a pas de groupement de cellules visible sans passage à une forme primitive. Il y a trois états sources a, b et j. La réalisation de cet automate nécessite 10 cellules, dans la methode initiale.

lère étape.

Dans la première étape on détermine une représentation de l'automate par un ensemble plus ou moins minimisé de variables. Ici nous utilisons les méthodes de DAVID pour obtenir la table des variables suivantes :

000	001	011	010	110	101	101	111	$r_1 r_2$
0	(y_a)	(y_a)	y_f	∅	∅	y_d	0	01
(y_b)	0	∅	y_e	∅	y_i	∅	0	00
0	∅	∅	y_f	(y_c)	(y_c)	y_d	0	01
0	0	∅	∅	y_c	(y_d)	(y_d)	0	01
0	∅	0	(y_e)	(y_e)	y_d	∅	0	11
0	0	(y_f)	(y_f)	y_c	∅	∅	0	01
0	∅	0	(y_g)	(y_g)	y_d	∅	0	10
0	0	∅	∅	y_g	(y_h)	(y_h)	0	10
0	∅	∅	∅	y_e	(y_i)	∅	∅	11
∅	∅	∅	∅	∅	∅	y_h	(y_j)	00

2ème étape.

1) En appliquant les règles de simplification du paragraphe 7-6, on obtient les consensus suivants d'ordre ≠ 0

$y_m = y_a \overset{*}{}_1 y_d =$	0	(y_m)	0	∅	∅	y_d	(y_m)	y_b	01
$y_n = y_f \overset{*}{}_1 y_c =$	0	∅	y_f	(y_n)	(y_n)	y_c	∅	y_b	01
$y_p = y_g \overset{*}{}_2 y_h =$	0	∅	∅	y_g	(y_p)	(y_p)	y_h, y_d	y_b	10

Ces trois variables sont nouvelles et sont adjointes à la table initiale.

On forme ensuite :

$y_q = y_a \underset{3}{\times} y_n =$

0	0	y_a	(y_q)	y_n	$\emptyset$	$\emptyset$	0	01

et cette variable est <u>équivalente</u> à y_f (car y_n est pseudo-équivalent à y_c.

De la même manière on constate que :

$$y_n \underset{2}{\times} y_d = y_c$$

$$y_c \underset{3}{\times} y_m = y_d$$

$$y_d \underset{2}{\times} y_f = y_a$$

2) Consensus d'ordre zéro :

$y_r = y_g \underset{0}{\times} y_n =$

0	$\emptyset$	y_f	(y_r)	(y_r)	y_c, y_d	$\emptyset$	0	11

On constate que y_a remplace y_f pour la valeur X_3 = 011 et comme y_a est source, y_f sera simplement supprimé de $G(y_r, X_3)$.
De même y_d remplace $\{y_c, y_d\}$ pour la valeur X_6 = 100 et y_r peut être simplifié en

$y_r =$

0	$\emptyset$	0	(y_r)	(y_r)	y_d	$\emptyset$	0	11

Et on constate que y_r est équivalent à y_e

$y_s = y_h \underset{0}{\times} y_d =$

0	$\emptyset$	$\emptyset$	$\emptyset$	y_c, y_g	(y_s)	(y_s)	0	11

Ici, y_n remplace y_c pour la valeur X_5 = 110 et y_s peut s'écrire :

$y_s =$

0	$\emptyset$	$\emptyset$	$\emptyset$	y_a, y_g	(y_s)	(y_s)	0	11

Mais $\{y_n, y_g\}$ remplace y_c donc y_s remplace y_i (il n'est pas équivalent puisque y_s est stable en un point où y_i ne l'est pas).
Les autres consensus d'ordre zéro ne donnant aucun remplacement, ils sont donc inutiles à la suite de la méthode.

Troisième étape.

Nous cherchons une couverture minimale des variables initiales : nous développons donc l'expression :

$(z_a + z_d \cdot z_f)(z_b)(z_c + z_n z_d)(z_d + z_c z_m)$.

$(z_e + z_g \cdot z_n)(z_f + z_a \cdot z_n)(z_g)(z_h)(z_i + z_d z_h)$

dont chaque terme correspond à une relation de remplacement par un consensus.

En développant on trouvera en particulier le terme

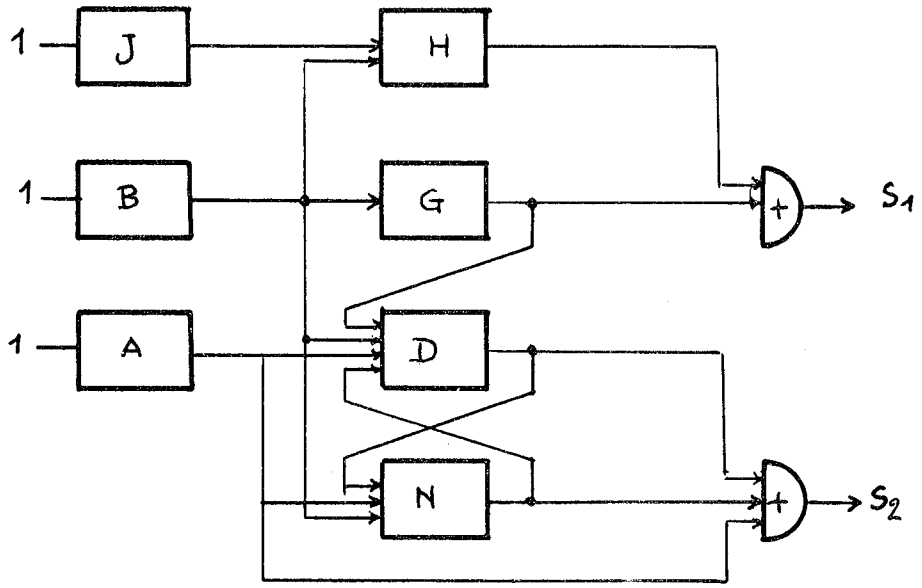
$$z_a \cdot z_b \cdot z_c \cdot z_d \cdot z_e \cdot z_f \cdot z_g \cdot z_h \cdot z_i$$

qui correspond à la couverture initiale. Mais on trouve également le terme $z_a \cdot z_b \cdot z_d \cdot z_g \cdot z_h \cdot z_n$

qui correspond à la représentation suivante :

000	001	011	010	110	100	101	111	$r_1 r_2$
0	(y_a)	(y_a)	y_n	∅	∅	y_d	0	01
(y_b)	0	∅	y_g, y_n	∅	y_h, y_d	∅	∅	00
0	0	∅	∅	y_n	(y_d)	(y_d)	0	01
0	∅	0	(y_g)	(y_g)	y_d	∅	0	10
0	∅	∅	∅	y_g	(y_h)	(y_h)	0	10
0	∅	0	(y_n)	(y_n)	y_d	∅	0	01
∅	∅	∅	∅	∅	∅	y_h	(y_j)	00

et le réseau correspondant est le suivant :



CONCLUSION.

Cet exemple montre bien le gain qui peut être obtenu en utilisant la méthode des consensus puisque d'un réseau initial à 10 cellules on a pu passer sans revenir à la forme primitive à un réseau à 7 cellules.

C H A P I T R E 8

ETUDE DES SORTIES

8 - 1 Remarque 1.

Une étude approfondie de la méthode de conception de DAVID montre que celle-ci a deux aspects :

- Elle est intéressante tout d'abord parce que le codage des états et les éléments de mémoire utilisés permettent de réduire l'étude séquentielle.

- Elle est bien adaptée à résoudre le problème combinatoire du calcul des sorties, puisque celles-ci sont réalisées pratiquement de manière canonique.

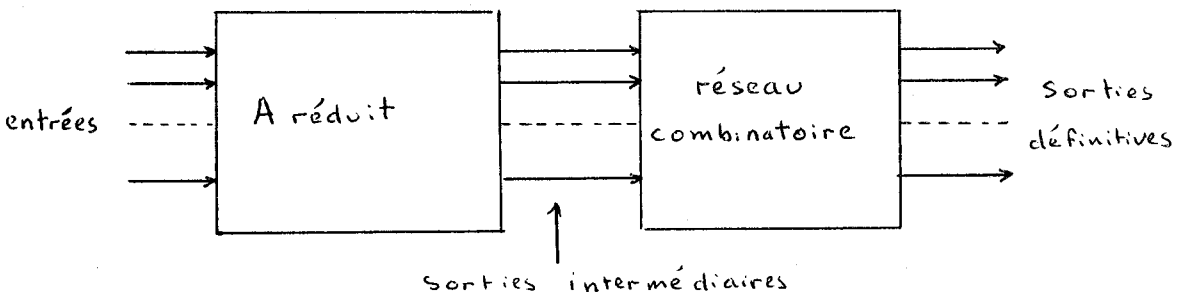
Ces deux aspects sont en fait d'un intérêt très différent. Le premier est essentiel : que le codage se fasse manuellement ou par ordinateur, le problème est complexe et mal résolu jusqu'ici. La méthode de DAVID apporte ici une solution simple et intéressante. Par contre, en ce qui concerne l'élaboration des sorties, le problème est totalement différent : si cette partie est traitée en ordinateur, de nombreuses simplifications supplémentaires peuvent être apportées. Par ailleurs, si la cellule CUSA est un élément très simple en logique intégrée, il n'en est pas de même des opérateurs OU utilisés pour les sorties. En logique intégrée on rencontre plutôt le NOR (en M.O.S.) ou le

NAND (en T.T.L.). La différence essentielle étant que ces opérateurs ne sont pas associatifs et qu'on ne peut, par exemple, pas faire un NOR à 5 entrées avec deux NOR à 3 entrées.

Remarque 2.

Dans les méthodes précédentes toutes les réductions sont basées sur l'existence de relations d'hypovalence. Toutefois la probabilité de trouver de telles relations est faible dès qu'il y a plusieurs sorties : il suffit par exemple qu'une sortie soit le complément d'une autre pour qu'aucune relation n'existe.

On est ainsi amené à reconsidérer le problème de l'élaboration des sorties. On s'efforcera de définir une première série de "sorties intermédiaires" permettant de faire apparaître le nombre maximum d'hypovalence, et servant de base à un réseau combinatoire complexe réalisant les sorties définitives.



Deux méthodes peuvent être étudiées.

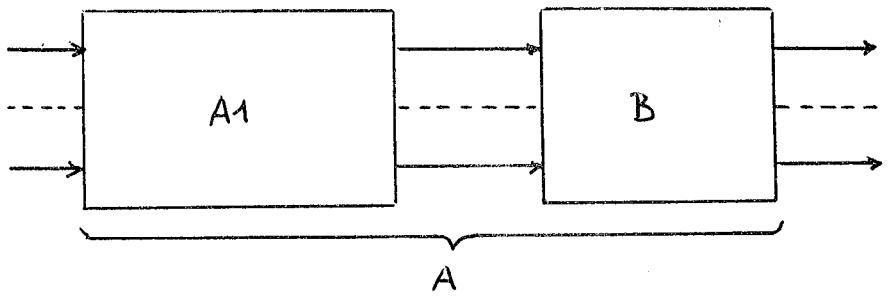
1) Une méthode plus algébrique permettrait uniquement de définir le codage optimal des sorties pour avoir le maximum de relations.

2) Une méthode topologique tenant plus compte des notions de coût, et en particulier du coût des connexions.

Ces deux méthodes sont dérivées de celles étudiées dans la première partie et pourraient être programmées, sensiblement, par les mêmes algorithmes.

Toutes deux font intervenir la même notion de base initiale du fractionnement, notion que nous allons développer ici.

8 - 2 Détermination de la base initiale.



On cherche quelles pourraient être les sorties de A1 pour que le nombre maximum de relations existent dans A1.

8-2-1 Définitions.

Hypovalence virtuelle.

Nous appellerons ainsi une $\emptyset$ relation entre états internes, itérante, non qualifiée sur les sorties, et non triviale (une hypovalence triviale étant par définition telle, que, pour tout couple d'états $\{a, b\}$ on ait $a \lhd b$ et $b \lhd a$).

Le terme virtuel provient du fait que si on donne aux sorties une valeur convenable, on peut arriver à transformer cette relation virtuelle en une relation d'hypovalence réelle.

Source virtuelle.

Un état sera dit source virtuelle s'il est hypovalent virtuellement à tous les états auxquels il est comparable.

Soit R une relation d'hypovalence virtuelle. Pour la rendre réelle, on doit définir des sorties S telles que pour tout couple $\{Y_i, Y_j\}$ de R avec $Y_i \lhd Y_j$, on ait $S(Y_i) \leq S(Y_j)$.

8-2-2 Algorithme de recherche d'une base.

A chaque état Y_i, on associe une sortie B_i telle que, pour tout $Y_p \triangleright Y_i$, on ait $B_i(Y_p) \geq B_i(Y_i)$

a) Si Y_i est maximal (pour la relation d'hypovalence), on prendra :

$$B_i = Y_i$$

b) Si Y_i n'est pas maximal, il existe $Y_p \triangleright Y_i$ et on doit prendre :

$$B_i \geq Y_i + Y_p$$

Si, pour tout $p \in P$ on a $Y_p \geq Y_i$, on prendra en définitive :

$$B_i = Y_i + \sum_{p \in P} y_p$$

et on aura bien $b_i(Y_p) \geq b_i(Y_i)$ pour tout p

8-2-3 Exemple.

Soit l'automate ci-dessous.

x_1x_2	00	01	11	10	S
(a)	b		f		01
c	(b)	e			11
(c)	d				10
c	(d)	e			00
	g	(e)	f		01
a		e	(f)		00
	(g)	h			01
	g	(h)			11

Les relations virtuelles sont :

$a \triangleleft c$ $\quad b \triangleleft d$ (1)

$b \triangleleft g$ $\quad e \triangleleft h$ (2)

$d \triangleleft g$ $\quad e \triangleleft h$ (3)

(ou leurs duales)

a, b, e, f sont sourcesvirtuelles.

Il n'y a ici aucune relation réelle.

a) Pour les états c, d, g, h, f on peut prendre

$B_c = y_c$, $B_d = y_d$, $B_g = y_g$, $B_h = y_h$, $B_f = y_f$

b) Si $B_a(a) = 1$, cela implique, puisque a < c, de prendre $B_a(c) = 1$ également

Si $B_b(b) = 1$ cela impose de prendre $B_b(d) = B_b(g) = 1$

Si $B_e(e) = 1$ cela implique $B_e(h) = 1$ et on est conduit à la base suivante :

	B_a	B_b	B_c	B_d	B_e	B_f	B_g	B_h	x_1	x_2
a	1									
b		1								1
c	1		1							
d		1		1						1
e					1				1	1
f						1			1	
g		1		1			1			1
h					1			1	1	1

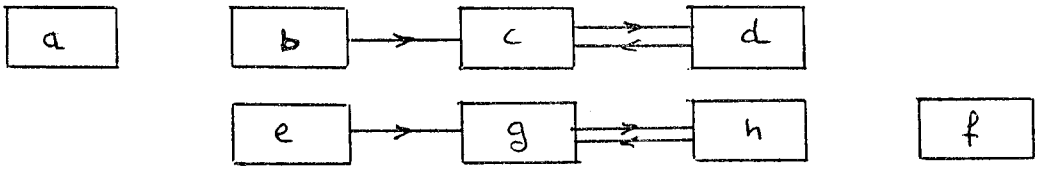
Remarque 1.

Il est à remarquer qu'on a rajouté à cette base les variables primaires x_1 et x_2 puisque rien n'interdit de les utiliser pour l'élaboration des sorties si cela simplifie la conception.

Remarque 2.

Dans une méthode topologique où il serait tenu compte du coût, on envisage la meilleure réalisation possible de l'automate.

Dans cet exemple on obtiendrait :



Dans cet exemple, a, b, e, f sont de coût 1.

c, d, g, h sont de coût 3 et évidemment, les variables, de coût 0.

8-2-4 Théorème.

Une base ainsi définie est suffisante pour la représentation de toute sortie.

En effet, on constate que toutes les "lignes" du tableau de la base sont distinctes et que toute fonction peut donc être représentée sur cette base.

Corollaire.

Toute relation virtuelle pourra être rendue réelle : on n'aura jamais besoin pour réaliser une fonction de sortie d'utiliser une fonction intermédiaire qui détruise les relations.

8-2-5 Choix des relations virtuelles.

On remarque qu'à toute relation R virtuelle correspond la relation R' complémentaire obtenue en inversant tous les couples de R. Il peut alors y avoir plusieurs manières de choisir parmi les relations virtuelles celles qui seront rendues réelles. On les choisira de manière à avoir le plus d'états source possible. Ceci se fera au moyen d'un très classique algorithme par dualisation.

Algorithme.

1) A toute relation R_i on associe une variable booléenne r_i égale à 1 si R_i est réelle, égale à zéro dans l'autre cas.

2) A tout état Y_i on associe une grandeur booléenne a_i égale à 1 si Y_i est source, égale à zéro s'il ne l'est pas. a_i peut être calculé : pour que Y_i soit source, il faut en général, qu'un certain nombre de relations R_{i1} R_{i2}.....R_{ik} soient réelles, donc

$$a_i = \prod_{m=1}^{k} r_{im}$$

(Si Y_i est seul dans sa colonne, $a_i = 1$)

3) Pour chaque colonne C_j on peut avoir

a) soit aucun état source, ce qu'on caractérise par une variable booléenne b_j

b) soit un état source parmi les Y_{ij} de la colonne C_j, ce qui peut être caractérisé par la grandeur :

$$\Sigma a_{ij}$$

Donc, dans chaque colonne C_j, on a :

$$b_j + \sum_i a_{ij} \equiv 1$$

4) Ceci étant vrai pour chaque colonne, on a :

$$\prod_j (b_j + \sum_i a_{ij}) \equiv 1$$

Si on développe cette expression, on obtient des monômes en a_{ij} et b_j, c'est-à-dire, en fait des monômes en b_j et r_m. Cherchant à avoir un nombre maximum d'états source, on cherche les monômes contenant un minimum de b_j.

a) On élimine les monômes nuls, c'est-à-dire ceux contenant

r_i attaché à R_i et r_{i1} attaché à R_i' puisqu'on ne peut avoir simultanément deux relations complémentaires.

b) On élimine un monôme M_1 devant un monôme M_2 si M_1 contient au moins tous les b_j de M_2 (si M_2 est caractéristique d'une combinaison plus petite d'états sources).

c) Enfin, parmi tous les monômes restant, on choisit un quelconque de ceux dont le nombre de b_j est le plus faible.

Exemple.

Dans l'exemple précédent, on trouve les relations virtuelles maximales suivantes :

R_1 : $\{a \lhd c , b \lhd d\}$ r_1

R'_1 : $\{a \rhd c , b \rhd d\}$ r_2

R_2 : $\{b \lhd g , e \lhd h , d \lhd g\}$ r_3

R'_2 : $\{b \rhd g , e \rhd h , d \rhd g\}$ r_4

Les variables attachées aux états sont :

$a_a = r_1$

$a_b = r_1 . r_3$

$a_c = r_2$

$a_d = r_2 . r_3$

$a_e = r_2$

$a_f = 1$

$a_g = r_4$

$a_h = r_4$

On a donc, pour chacune des colonnes

$$(b_1 + r_1 + r_2) \equiv 1$$

$$(b_2 + r_1r_3 + r_2r_3 + r_4) \equiv 1$$

$$(b_3 + r_3 + r_4) \equiv 1$$

$$(b_4 + 1) \equiv 1$$

On développera donc, en définitive, l'expression :

$(b_1 + r_1 + r_2)(b_2 + r_1r_3 + r_2r_3 + r_4)(b_3 + r_3 + r_4)$

sachant que $r_1r_2 = 0$ et $r_3r_4 = 0$

Il est inutile de tout développer. On constate que, par exemple, on trouve les termes

$$r_1r_4,\ r_2r_4,\ r_1r_3,\ r_2r_3$$

qui sont maximaux puisqu'ils correspondent à la présence d'un état source dans chaque colonne. Par exemple, les relations utilisées au paragraphe 8-2-3 étaient caractérisées par r_1r_3.

8 - 3 Méthode algébrique.

Toute sortie peut s'exprimer sous forme d'une fonction (incomplètement spécifiée) des variables primaires et de codage.

Cette fonction F(X, Y) peut être fractionnée par la méthode littérale, en utilisant comme base de départ celle définie précédemment.

8-3-1 Convergence.

La méthode de fractionnement converge vers la base si celle-ci au départ contient toutes les variables, ce qui n'est pas le cas ici. Si, par exemple, après un certain nombre de fractionnement on arrive à une fonction de la forme

$$f = \left[my_i,\ \mu y'_i\right]$$

et si $m.\mu = 0$, la seule solution est y_i et cette variable n'est pas forcément présente dans la base (si par exemple Y_i est source).

On est amené à considérer une base plus grande comportant toutes les

variables internes, mais chacune affectée d'un coût qui sera fonction du nombre de simplifications supprimées par son utilisation.

8-3-2 Exemple.

Soit un état Y_i source ayant P ascendants et vérifiant

$$Y_i \lhd Y_j \qquad (1)$$

On doit utiliser uniquement la fonction $s_i = y_i + y_j$ et non y_i seul si on désire garder (1).

Mais l'utilisation de y_i, si elle est obligatoire, nous fait perdre au moins la relation (1) et donc la propriété de Y_i d'être source. Le coût de y_i est donc au moins P.

Algorithme de synthèse.

Dans la recherche d'une solution d'une fonction $F = [\underline{f}, \overline{f'}]$ on n'explorera la partie de la base comportant des variables de coût élevé, que si F n'est plus fractionnable.

8 - 4 Méthode topologique.

8-4-1 Fermeture par antécédence.

Etant donné un état interne a, on considère l'ensemble F(a) des états ascendants immédiats ou non de a (sans tenir compte des ascendants des états source).

C'est le plus petit des ensembles $\mathcal{E}$ tels que

1) $a \in \mathcal{E}$

2) $b \in \mathcal{E} \iff \begin{cases} \forall X_k,\ G^{-1}(b, X_k) \in \mathcal{E} \\ \text{ou bien b est source} \end{cases}$

8-4-2 Fermeture maximale.

On appellera ainsi une fermeture non incluse dans une autre.

La fermeture de a est maximale si et seulement si les descendants de a

sont tous source.

Remarque.
Dans une réalisation de DAVID, les fermetures des états à sorties non nulles correspondent aux états représentés effectivement par une cellule.

Coût.
On appellera coût d'un état Y_i, le nombre d'états dans sa fermeture. Physiquement, c'est le nombre de cellules obligatoirement présentes si y_i est utilisé dans la réalisation des sorties.

8-4-3 Optimisation du coût.
Dans la méthode de DAVID on est amené à utiliser tous les états à sortie non nulle, et donc leur fermeture. Dans la méthode proposée ici, puisqu'on peut disposer des variables primaires, on cherchera à n'utiliser que les états de coût faible.

Remarque.
On voit ici l'intérêt de la notion d'hypovalence virtuelle : comme on ne sait pas à l'avance si un état sera ou non représenté par une cellule, tout se passe comme si on ne connaissait pas sa sortie.

8-4-4 Algorithme de synthèse.
Cet algorithme est déduit de celui relatif aux réseaux combinatoires ; la seule différence étant la notion de coût des éléments de la base qui intervient ici.

8-4-4-1 Fractionnement.

Etant données les fonctions de sorties f_1, f_2.....f_n définies sur la base :

$$\{x_1.....x_k, s_1.....s_n\}$$

étant donnés les coûts $\mathcal{C}_1 \ldots\ldots \mathcal{C}_k, \mathcal{C}_{k+1} \ldots\ldots \mathcal{C}_n$ des éléments de la base, les coûts $\mathcal{C}_i$, $1 \leq i \leq k$ étant nuls puisque l'utilisation d'une variable primaire ne coûte rien, on procède de la manière suivante.

a) Une première fonction f_i à réaliser est choisie.

b) On fractionne cette fonction f_i (suivant les méthodes combinatoires précédemment définies), jusqu'à obtention des fonctions résolubles par des éléments de la base. Si la méthode topologique est utilisée, on tiendra compte, non plus de la seule distance, mais du produit (coût) x (distance).

c) Quand une fonction intermédiaire f_p est résolue par un élément y_i de la base, on considère que le coût de cet y_i de sa fermeture est nul (puisque g_i est nécéssaire à f_p, il ne coûte rien de l'utiliser pour f_q).

d) Si une fonction intermédiaire n'est pas résolue par un élément de la base, on itère le procédé en (a).

e) On traite successivement chacun des f_i du problème.

8-4-4-2 Convergence.

Comme précédemment, la convergence vers la base initiale n'est pas assurée. Cependant, ici, on ne s'en remet pas au hasard pour trouver des fonctions dans la base; si l'on ne veut pas utiliser y_i, trop cher, mais $y_i + y_p$ qui ne détruit pas de relations, on peut s'imposer, d'entrée, d'utiliser cette fonction. Autrement dit, on peut s'imposer que le pivot soit choisi parmi certains éléments de la base, ceux qui ne détruisent pas de relations.

Remarque.

On ne démontre pas que si $y_i + y_p$, par exemple, est imposé comme pivot au premier pas du fractionnement, (parce que y_i utilisé seul détruirait une relation) on ne retrouve pas plus avant dans le fractionnement, une fonction f_i ayant comme solution y_i. Mais, ici, contrairement à ce qui

se passait dans la méthode littérale, la convergence n'impose pas forcément la présence de y_i, car on peut fractionner une fonction jusqu'à n'avoir plus qu'un seul point de spécification à la fois.

CHAPITRE 9

DECOMPOSITION DES AUTOMATES

Introduction.

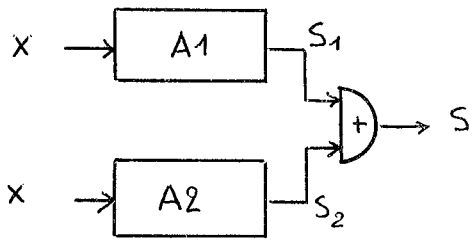
Dans les méthodes nouvelles de fabrication de circuits, l'intégration à grande échelle prend de plus en plus d'importance, mais actuellement seule la moyenne échelle peut être abordée de manière courante. Cela signifie que les automates trop "gros" pour tenir sur une puce de taille raisonnable, au regard du rendement et de la fiabilité, ne pourront pas être intégrés s'ils ne sont pas décomposés. On sait, depuis longtemps, par la méthode d'Hartmanis décomposer certains automates, mais on peut faire à la méthode les critiques suivantes.

- La majorité des automates ne sont pas décomposables au sens d'Hartmanis Si certains automates à fonction mathématique utilisés dans les ordinateurs sont décomposables, il est très rare qu'un système industriel le soit.
- La décomposition de Hartmanis minimise le nombre de variables intervenant dans les diverses parties composant l'automate. Ce critère est purement mathématique et ne correspond nullement à un critère de plus grande simplicité des réseaux. En fait, pour la plupart des constructeurs, c'est le nombre d'interconnexions entre puces qui fait pratiquement le prix de l'automate total, c'est ce nombre qui doit être minimisé.

9 - 1 Définition : décomposition en somme ou produit.

On appelle somme de deux automates l'automate obtenu en faisant l'union des sorties de ces automates (il y a plusieurs sommes possibles pour des automates à plusieurs sorties).

Exemple.



Nous cherchons à décomposer un automate en somme d'automates plus simples (ou en produits d'automates).

Nous allons exposer cette méthode dans le cas d'automates à une seule sortie mais le principe est valable dans le cas général.

Etant donné un automate A dont la sortie est S, nous définissons deux automates A1 et A2 ayant même tableau d'états et des sorties S1 et S2 telles que

$$S1 + S2 = S$$

Soit L l'ensemble des lignes du tableau d'état A pour lesquelles la valeur de S est 1. Nous définissons une partition P = {L1, L2} de L. Sur toute ligne de L1, la valeur de S1 sera 1 et celle de S2 sera 0 ou 1 (valeur indéterminée notée $\emptyset$) ; de la même manière sur toute ligne de L2, la valeur S1 sera $\emptyset$ et celle de S2 sera 1. On obtient ainsi deux automates plus incomplets que A et qui seront aisément

réductibles.

9 - 2 Exemple

A =

00	01	11	10	S
(a)	b	-	d	0
a	(b)	c	-	1
-	f	(c)	d	0
e	-	g	(d)	1
(e)	f	-	d	1
e	(f)	c	-	0
-	b	(g)	-	0

Dans cet automate seul est source l'état d et il est indispensable puisqu'à sortie non nulle ; il faut donc 7 cellules pour le réaliser, sans tenir compte ici d'éventuelles possibilités de groupement.
On définit alors A1 et A2 par la partition de la sortie S

A1 =

(a)	b	-	d	0
a	(b)	c	-	1
-	f	(c)	d	0
e	-	g	(d)	∅
(e)	f	-	d	∅
e	(f)	c	-	0
-	b	(g)	-	0

A2 =

(a)	b	-	d	0
a	(b)	c	-	∅
-	f	(c)	d	0
e	-	g	(d)	1
(e)	f	-	d	1
e	(f)	c	-	0
-	b	(g)	-	0

Du fait de l'apparition de valeurs ∅, certaines relations d'hypovalence vont apparaître.

Dans A1 on a en effet

e hypovalent à a

f " b

c " g

Donc les états e, f, c, d sont sources. On constate alors que les états c, e, f n'ont pas à être représentés.

De même dans A2 on a les relations suivantes :

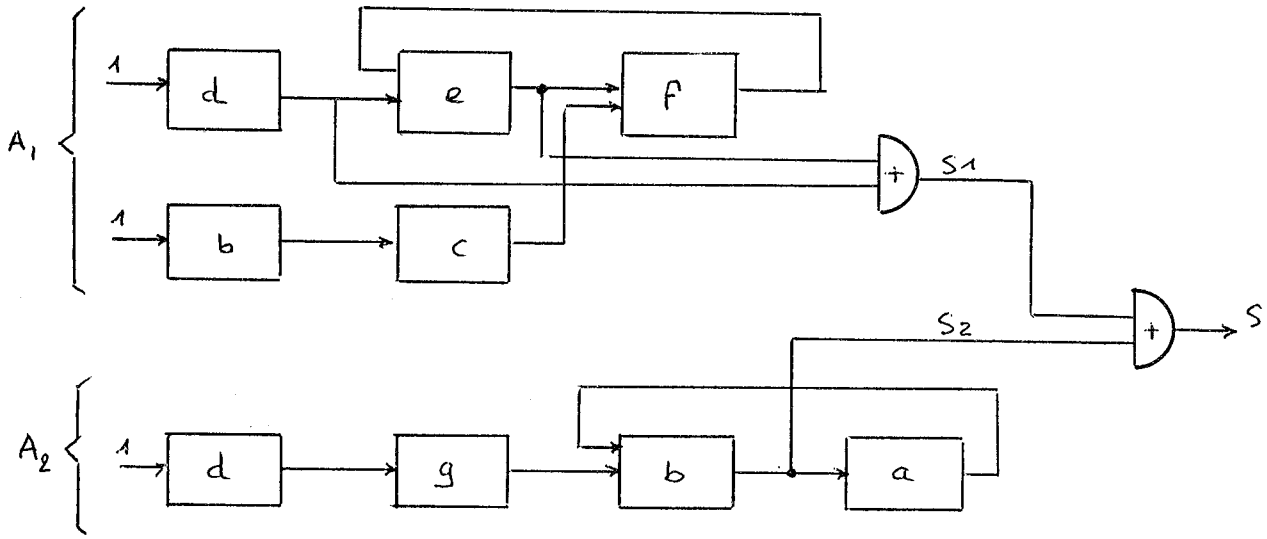
a hypovalent à e

b " f

g " c

Donc ici a, b, g, d sont sources et on constate que cette fois a et g n'ont pas à être représentés.

On est conduit à la réalisation suivante :



Au prix, ici, de 2 cellules supplémentaires, on a gagné une décomposition.

9 - 3 Etude mathématique.

9-3-1 Définition.

1) Etant donnée une relation virtuelle R qui n'est pas réelle, on appelle couple inhibant de R un couple $\{S(y_i)\ S(Y_j)\}$ de valeurs de sorties correspondant aux états Y_i et Y_j, et liées par une relation incompatible avec celle liant Y_i et Y_j dans R.

Si, dans R on a $Y_i \lhd Y_j$, on aura donc :

- soit $S(Y_i) > S(Y_j)$
- soit $S(Y_i)$ et $S(Y_j)$ non comparables.

2) On appelle relation virtuelle stricte une relation d'hypovalence qui ne contient pas de sous relation d'équivalence virtuelle.

Si une relation R est stricte, elle est disjointe de son complément R' (aucun couple ne figure dans R et dans R').

9-3-2 Décomposition en sommes.

Lemme 1.Etant donné un automate A de vecteur de sortie S et de graphe G, et deux automates A1 et A2 de même graphe G et de sorties S_{A1} et S_{A2} vérifiant

$$S_{A1} + S_{A2} = S$$

Alors l'automate somme de A1 et de A2 est équivalent à A si les états initiaux sont identiques.

Appelons $V_1\ V_2 \ldots\ldots V_n$ les états de A1 et $W_1\ W_2 \ldots\ldots W_n$ les états correspondant de A2. A partir du couple d'état initial $V_1\ W_1$ on est conduit à des couples d'états correspondants puisque A1 et A2 ont même graphe. L'ensemble des états globaux est donc formé uniquement de couples d'états $(V_1\ W_1)$, $(V_2\ W_2) \ldots\ldots (V_n\ W_n)$. Par ailleurs, les sorties ayant la bonne valeur à cause de la condition sur les sorties, on a bien un automate équivalent à A.

Propriété.

La propriété d'équivalence entre les automates reste vraie si on réduit séparément chacun d'eux.

En effet, bien que la règle de formation des états devienne complexe, on conçoit que, puisque les sorties des automates A1 et A2 ne sont pas changées par la réduction, leur somme n'est pas modifiée, d'où la propriété.

9-3-3 Décomposition d'après une relation stricte.

Soit A un automate, R une relation virtuelle stricte sur A et R' son complément. Nous supposerons que ni R ni R' ne sont réelles (sinon le problème est déjà résolu).

Nous allons construire deux automates B et C ayant les mêmes transitions que A (même tableau d'état), et des sorties T et U vérifiant pour tout état Y_i

$$T(Y_i) + U(Y_i) = S(Y_i)$$

Nous imposerons évidemment que B et C soient plus simples que A, en imposant ici que R soit réelle dans B et R' réelle dans C.

Puisque R n'est pas réelle dans A, il existe au moins un couple inhibant $\{S(Y_i), S(Y_j)\}$; supposons que dans R on ait:

$$Y_i \lhd Y_j$$

Deux cas peuvent se présenter

a) $S(Y_i) > S(Y_j)$

Le couple est donc alors seulement inhibant pour R (mais pas pour R'). On peut définir alors T et U par

$$T(Y_i) = T(Y_j) = S(Y_j)$$

$$U(Y_i) = S(Y_i),\ U(Y_j) = S(Y_j)$$

et on aura détruit le couple inhibant pour R dans B tout en ayant

$$S(Y_i) = T(Y_i) + U(Y_i)$$
$$S(Y_j) = T(Y_j) + U(Y_j)$$

b) $S(Y_i)$ et $S(Y_j)$ ne sont pas comparables : le couple est inhibant pour R et pour R'. On définit T et U par

$$T(Y_i) = S(Y_i) \cdot S(Y_j)$$
$$T(Y_j) = S(Y_j) \qquad \text{on a ainsi } T(Y_i) \leq T(Y_j)$$
$$U(Y_j) = U(Y_i) \cdot U(Y_j)$$
$$U(Y_i) = S(Y_i) \qquad \text{on a ainsi } U(Y_i) \geq U(Y_j)$$

et les sorties vérifient donc

$$T(Y_i) + U(Y_i) = S(Y_i) \cdot S(Y_j) + S(Y_i) = S(Y_i)$$

De même pour Y_j.
On a donc détruit un couple inhibant pour R et pour R'.

Itération de la méthode : convergence.

On cherchera à détruire tous les couples inhibant de R et R', mais peut-on faire ceci sans créer de nouveaux couples ? Autrement dit, la méthode converge t-elle ?

a) Dans le cas général, la destruction d'un couple peut entraîner la création d'autres couples : par exemple si dans R on trouve trois états vérifiant

$$Y_i \lhd Y_k,\ Y_j \lhd Y_k$$

et si on a :

$$S(Y_i) = 001 \quad S(Y_j) = 010 \quad S(Y_k) = 110$$

le couple $\{S(Y_j)\ S(Y_k)\}$ n'est pas inhibant mais $\{S(Y_i), S(Y_k)\}$ est inhibant.
On est dans le cas (b) et on définit donc T la sortie de B par :

$$T(Y_i) = S(Y_i),\ T(Y_k) = S(Y_i) \cdot S(Y_k)$$

On aura donc

$$T(Y_i) = 001 \quad T(Y_j) = 010 \quad T(Y_k) = 000$$

et on aura créé un couple inhibant nouveau.

b) Convergence. On montre cependant que la méthode converge : à chaque destruction d'un couple inhibant, la sortie de B diminue par rapport à celle de A et donc (le nombre de valeurs de sorties étant fini), la méthode converge toujours.

c) Remarque. On a supposé l'hypovalence stricte, c'est-à-dire qu'on n'a pas dans R un couple $Y_i \lhd Y_j$ et $Y_j \lhd Y_i$.
Supposons alors que R n'est pas stricte : il existe une sous relation R_1 de R qui est une relation d'équivalence. Deux cas peuvent se présenter.

- Il n'y a pas de couple inhibant sur R1. Donc R1 est réelle et l'automate A peut, dès le départ être réduit.

- Il y a un couple inhibant dans R1 : $\{S(Y_i), S(Y_j)\}$. Ce couple est inhibant pour R et pour R'. On s'efforce de réaliser dans B et dans C :

$$Y_i \lhd Y_j \text{ et } Y_j \lhd Y_i \quad \text{donc} \quad S(Y_i) = S(Y_j)$$

Mais ceci étant réalisé dans B et dans C, la somme T + U des sorties de B et C ne pourra pas être égale à S puisque

$$S(Y_i) \neq S(Y_j) \text{ et } T(Y_i) + U(Y_i) = T(Y_j) + U(Y_j)$$

9-3-4 Choix des relations.

Comme dans le chapitre précédent il se pose un problème de choix des relations à rendre réelles : la multiplicité des relations ne garantit nullement la simplicité de la réalisation.
Comme précédemment donc, on choisira de rendre réelles les relations qui permettent d'avoir un nombre maximum d'états sources. Cependant ici on doit avoir le nombre maximum dans B et dans C et on pourra chercher parmi les diverses solutions possibles, un couple de solutions impliquant des relations complémentaires.
Exemple : au paragraphe 8-2-5 on trouve comme solutions optimales les ensembles de relations

$$r_1 r_4,\ r_2 r_4,\ r_1 r_3,\ r_2 r_3$$

avec r_1 complémentaire de r_2

r_3 " r_4

On choisira par exemple pour relations virtuelles r_1 et r_4 dont le complémentaire donne encore un état source dans chaque colonne.

9-3-5 Conclusion.

La méthode est intéressante car elle donne des résultats dans un grand nombre de cas où la méthode d'Hartmanis ne donne pas de solution : la méthode d'Hartmanis suppose l'existence de relations d'équivalence, ce qui est un cas plus restrictif que les relations d'hypovalence.

B I B L I O G R A P H I E

Nous donnerons ici une bibliographie des ouvrages et articles en logique combinatoire ou séquentielle ; seront présentés les travaux qui sont considérés comme faisant autorité dans le domaine de la logique, (soit qu'ils présentent des résultats originaux utiles à la conception assistée, soit qu'ils réalisent une mise à jour des connaissances dans le domaine de la logique). Les travaux présentés ici seront classés en deux rubriques : une première relative à la logique dite combinatoire, et une deuxième réservée aux problèmes séquentiels.

I - Logique combinatoire.

Dans ce domaine les travaux sont extrêmement nombreux , mais très peu abordent le problème sous l'angle de la conception assistée. Nous distinguerons les travaux théoriques traitant en général de l'Algèbre de Boole et les travaux plus spécialisés abordant réellement le problème de la conception.

I-A. Ouvrages généraux.

[1] J. KUNTZMANN
"Algèbre de Boole"
DUNOD 1968

Ce livre fait une synthèse complète des méthodes théoriques appliquées au problème de la conception des circuits logiques. Présenté sous une forme mathématique simple, il aborde les problèmes sous leur aspect algorithmique. En outre ce livre est un des rares qui ne limite pas le problème à la minimisation des fonctions écrites sous forme polynomiale ; on trouvera en particulier, outre cette théorie, des méthodes de minimisation des fonctions incomplètes, de plusieurs fonctions, simultanément, des méthodes d'écriture sous forme galoisienne, et surtout les méthodes de base de synthèse à l'aide d'opérateur croissants et monotones.

[2] P. TISON

"Théorie du consensus"

Thèse de docteur-ingénieur. Grenoble 1965.

Cette étude contient une théorie de la minimisation des formes polynomiales par la méthode des consensus ; Elle est présentée sous un aspect appliqué. Elle est résumée sous une forme plus théorique dans 1 .

[3] F. LAPSCHER

"Application de la notion de fermeture à l'étude des fonctions booléennes"

Thèse de doctorat. Grenoble 1968.

Comme son titre l'indique, cette thèse présente des outils mathématiques utiles à l'étude des fonctions booléennes, et donc à la synthèse de ces fonctions. Ces outils sont à notre avis indispensable dans certains cas, en particulier pour l'étude des fonctions incomplètes et nous les avons largement utilisés.

[4] E. PICHAT

"Contribution à l'algorithmique non numérique dans les ensembles ordonnés"

Thèse de doctorat. Grenoble 1970

Cette thèse traite de plusieurs problèmes intervenant dans la conception, sous l'angle des méthodes de calcul. Son grand mérite est de donner non seulement une théorie, mais des méthodes de calcul applicables sur ordinateur, et dont l'efficacité a été démontrée et vérifiée. Parmi les problèmes traités on relève: des méthodes de décomposition (chapitre 1 et 2), des méthodes généralisant la notion de consensus (chapitre 3) dont nous nous sommes servis dans notre thèse, et enfin (au chapitre 6) des algorithmes rapides de couverture.

I-B. Conception des réseaux combinatoires.

Nous trouvons dans cette rubrique deux types de travaux : des livres en général destinés aux étudiants et ingénieurs, et des ouvrages de recherche plus spécialisés.

(1) Ouvrages généraux abordant la conception.

[5] CALDWELL S.H.
"Switching circuits and logical design"
John Wiley 1958.

Ce livre a été pendant longtemps considéré comme l'ouvrage de base pour la conception des réseaux, on y trouve les principes de base de la synthèse des circuits logiques tant combinatoires que séquentiels, (bien que ce dernier point soit moins développé). Actuellement ce livre est dépassé en raison de l'évolution technologique due aux circuits intégrés qui a déplacé les problèmes, et également en raison de l'évolution des méthodes mathématiques.

[6] Mc. CLUSKEY E.J.
"Introduction to the synthesis of switching circuits"
Mc. Graw Hill 1965.

Mc. CLUSKEY est un des premiers auteurs ayant abordé le problème de la minimisation des fonctions logiques sous son aspect algorithmique, et en vue d'un calcul automatique. Cet ouvrage présente entre autre la méthode qui porte son nom, et également celle des consensus. Ce livre est essentiellement destiné aux étudiants et ingénieurs et ne présente pas de méthodes d'avant garde en conception assistée.

[7] NASLIN P.
"Circuits logiques et automatismes à séquences"
Ce livre présente toutes les méthodes de la conception assistée sous une forme intuitive directement utilisable par des ingénieurs, mais n'aborde pas le problème de la conception assistée.

[8] PERRIN J.P., DENOUETTE M., DACLIN E.
"Systèmes logiques" Tomes 1 et 2.
DUNOD 1967.
Ce livre destiné aux ingénieurs et chercheurs est consacré essentiellement aux méthodes séquentielles. La logique combinatoire y est néanmoins traitée en tenant compte des divers critères imposés par les technologies récentes. Comme les précédent ce livre n'aborde pas les problèmes de synthèse par calculateur.

[9] LAGASSE J.
"Logique combinatoire et séquentielle"
DUNOD 1969
Ce livre est destiné essentiellement aux étudiants et, à ce titre, se limite aux problèmes relativement classiques de minimisation des fonctions écrites sous forme polynomiale.

[10] VALLEE R.L.
"Analyse binaire" Tome 1
MASSON & Cie 1970

Le premier tome de ce livre est consacré à l'algèbre des systèmes combinatoires, sous un formalisme qui sort des sentiers battus. Les notations utilisées donnent dans certains cas des résultats très intéressants : (les expressions logiques sont écrites sous une forme bidimensionnelle qui se rapproche parfois beaucoup de la topologie des réseaux). La notation est voisine de celle utilisée par ACKERS [12] et nous-mêmes [14] pour la synthèse des fonctions incomplètes.

(2) Articles specialisés dans la conception.

[11] MEO A.R.
"Sulla sintesi di reti NAND o NOR a molti livelli"
CALCOLO 1964

Cet article présente deux méthodes. Une première utilise une extension de la notion de composants premiers par exemple des produits de sommes de variables (au lieu de produits de variables). Cette méthode est rigoureuse mais totalement inapplicable même avec trois niveaux en raison du grand nombre de termes qu'il faut considérer. La deuxième méthode heuristique mais plus réaliste utilise des mises en facteurs dans des formes polynomiales minimale. Les deux méthodes ne sont généralisables ni aux fonctions incomplètes, ni au cas de plusieurs fonctions.

[12] ACKERS S.B.
"A diagrammatic approach to multilevel logic synthesis"
IEEE trans. EC 14 n° 2 pp. 174-181- 1965

Cet article est un des premiers permettant d'aborder de manière rigoureuse et efficace le problème de la synthèse, au moyen d'opérateurs modernes (NOR et NAND), de fonctions incomplétement spécifiées. Cet article a inspiré une partie de notre thèse de docteur-ingénieur et le chapitre trois de notre présente thèse. Néanmoins ACKERS n'abordait le problème que sous un aspect graphique sans envisager une synthèse automatisée.

[13] LUSTMAN F.
"Réalisation de fonctions booléennes avec l'opérateur Majorité"
Thèse de docteur-ingénieur. Grenoble 1966.

Cette thèse constitue un des premiers travaux de conception assistée à l'aide d'opérateurs autres que ET et OU (et bien sur NAND et NOR qui en dérivent). La méthode proposée est intéressante pour des fonctions dépendant de peu de variables, dans ce cas elle permet d'obtenir le réseau rigoureusement minimum. Dans le cas plus général, elle doit être simplifiée pour donner diverses méthodes heuristiques qui sont actuellement dépassées. La critique majeure qu'on peut faire à ces méthodes est de n'être pas du tout adaptées au problème de la synthèse simultanée de plusieurs fonctions

[14] DESCHIZEAUX P.
"Synthèse des fonctions booléennes générales"
Thèse de docteur-ingénieur. Grenoble 1967.
Dans cette thèse nous avions développé une méthode de fractionnement littérale s'apparentant à celle d'ACKERS [11] , et comparé son efficacité avec les méthodes proposées par LUSTMAN [12] .

[15] CHEIN
"Etude des décompositions d'un réseau, applications à l'écriture des fonctions booléennes en sommes et produits"
Thèse de docteur-ingénieur. Grenoble 1967.
Cette étude reprend nos travaux cités en [14] , en l'améliorant par l'apport d'une méthode de décomposition des fonctions booléennes à l'aide d'une méthode de graphes.

[16] MORREALE E.
"Partitioned list algorithmes for prime implicants determination from canonical forms"
IEEE trans. EC 16, pp. 611-620, 1967
Bien que la méthode proposée ne soit pas réellement une méthode de synthèse mais un outil pour la synthèse, elle mérite d'être mentionnée : elle est conçue pour les ordinateurs et elle est la plus rapide des méthodes actuelles.

[17] GIMPEL J.F.
"The minimisation of TANT Networks"
IEEE trans., EC 16, pp. 18-38 1967.

Cette méthode de synthèse par des NAND est une extension des méthodes de composants premiers, et, à ce titre, relativement peu efficace dès que le nombre de couches logiques devient supérieur à trois. Elle est à rapprocher des méthodes de MEO [11].

[18] SCHNEIDER P.R., DIETMEYER D.L.
"An algorithm for synthesis of multiple output combinatorial logic"
IEEE Trans., C 17, pp. 117-128, 1968

Cette étdue envisage la synthèse par des procédés de décomposition. La méthode est donc relativement complexe, et ne donne de bons résultats que pour des fonctions très incomplètes (pour lesquelles la probabilité de décomposition est grande). Nous notons toutefois une tendance à utiliser des modules complexes, et en particulier ceux que nous proposons au chapitre 3. Cet article est commenté au [18 bis].

[18 bis] DAVIDSON E.S., METZE G.
"Comment on an algorithm for synthesis of multiple output combinatorial logic"
IEEE Trans., C 17, pp. 1091-1092, 1968

[19] DIETMEYER D.L., SU Y.H.
"Logic design automation of fan-in limited NAND networks"
IEEE Trans., C 18, pp. 11-21, 1969

Cette méthode utilise un procédé de mise en facteur dans des expressions polynomiales ; Elle donne des circuits arborescents relativement coûteux à plusieurs sorties.

[20] DAVIDSON E.S.
"An algorithm for NAND decomposition under networks constraints"
IEEE Trans. C 18, pp.1098-1109 - 1969

Cet article présente une méthode programmée de synthèse de réseaux réels en tenant compte des contraintes technologiques. Bien que mathématiquement les méthodes diffèrent, l'esprit de cet article est voisin de celui de notre thèse : Il ressort clairement que le problème n'est plus dans le choix d'une méthode de synthèse logique, mais dans l'adaptation de la méthode aux divers problèmes annexes.

[21] CHAKRABARTI K.K., CHOUDHURY A.K., BASU M.
"Complementary fonction approach to the synthesis of three level NAND networks"
IEEE Trans., C 19, pp. 508-514, 1970

Comme [11] et [17] cet article étend la notion d'implicants premiers et la même remarque peut être faite à son sujet.

[22] BERTRAND J.C.
"Application des matrices booléennes à la synthèse modulaire des fonctions logiques"
Thèse de 3ème cycle TOULOUSE 1970.

Cette étude qui est très intéressante sur le plan théorique ne semble pas immédiatement utilisable sur ordinateur. La méthode qui est dans sa version initiale voisine de celle de LUSTMAN [13] conduit à un réseau rigoureusement minimum, par un algorithme un peut lourd qui ne peut être utilisé que pour des fonctions très simples. Par contre une deuxième méthode, heuristique, permet de traiter des problèmes plus importants ; elle utilise une méthode de décomposition voisine de [15] .

[23] MONTAGNON J.A.
"Synthèse booléenne dans les réseaux cellulaires"
Thèse de docteur-ingénieur. Grenoble 1971.

Cette thèse présente dans sa première partie une étude intéressante des structures régulières de réseaux cellulaires, qui présente toutefois l'inconvénient de ne pas toujours donner de solution. On peut regretter qu'une étude des fonctions synthétisables ou non, n'aie pas été faite. La deuxième partie utilise le principe de la synthèse à l'aide d'opérateurs croissants développée par ACKERS et qui peut être trouvée dans [1] et [13] .

Conclusion de cette bibliographie

On sent une très nette évolution des travaux en regardant les articles publiés ces dernières années : il y a dix ans les travaux abordaient essentiellement des méthodes mathématiques de traitement des fonctions logiques ; il y a cinq ans, la majorité des travaux traitaient des problèmes de synthèse à l'aide d'opérateurs très divers. Actuellement l'axe de recherche essentiel est la conception dans son ensemble compte tenu de contraintes variées, en particulier de contraintes d'implantation. Actuellement, une bonne méthode de synthèse est une méthode qui permet non pas de minimiser des critères de coût très complexes et très souvent le problème n'est plus dans la théorie mais dans l'énoncé du problème à résoudre et la définition des critères à minimiser.

I-C Divers.

[24] SELLERS F.F., HSIAO M.Y., BEARNSON L.W.
"Analyzing errors with boolean difference"
IEEE Trans., C 17, pp. 676-683 1968.

Cet article utilise la dérivée booléenne pour l'analyse des pannes aux entrées des réseaux. L'étude est relativement succinte, présente plus des idées générales qu'une méthode proprement dite.

[25] ACKERS S.B.
"On a theory of Boolean Functions"
J. SIAM VOL. 7 - Décembre 1959

Cet article présente en particulier la théorie des dérivées booléennes utilisées dans notre thèse.

II - Systèmes séquentiels.

Dans le domaine séquentiel on peut encore trouver deux sortes de travaux : théoriques et appliqués. La partie théorique est très abondante, nous ne présenterons que les ouvrages traitant de la théorie des automates en vue d'une réalisation électronique, en passant sous silence toute la partie concernant les théories qui abordent le problème sous l'angle de la théorie des langages.

II-1. Théorie des systèmes séquentiels.

Les ouvrages traitant de la théorie des automates sont relativement récents. Sans remonter à CALDWELL [5] , on peut citer :

[26] HARRISSON M.A.
"Introduction to switching and automata theory"
Mc. Graw Hill 1965.

Ce livre qui est en principe destiné aux ingénieurs, aborde toutefois le problème de la synthèse des machines séquentielles sous un aspect relativement mathématique. Il présente en effet outre les méthodes traditionnelles, le principe des expressions régulières et les machines non déterministes qui sont utiles aux synthèses telles que nous les envisageons.

[27] HARTMANIS J., STEARNS R.E.
"Algebric structure theory of sequential machine"
Prentice Hall, inc. 1966.

Ce livre est considéré en général comme l'ouvrage fondamental sur la théorie des machines séquentielles. Présenté sous un aspect purement algébrique, il développe essentiellement la théorie des partitions dans les systèmes séquentiels, et son application aux décompositions.

[28] TOU J.T.

"Applied automata theory"

Academic press 1968.

Ce livre écrit avec la participation de plusieurs auteurs présente des résultats très disparates. On y trouve en particulier :

- Une théorie succinte des expressions régulières, et de leur application à la description des graphes orientés.
- Une théorie des automates non déterministes dont le formalisme peut être utilisé à résoudre des problèmes de synthèse de certains systèmes déterministes.
- Une originale étude de machines cellulaires.

L'ensemble constitue plutôt un ouvrage de recherche qu'un livre destiné immédiatement à la conception.

II-2 Conception des systèmes séquentiels.

Nous ne donnerons dans cette rubrique qu'un petit nombre d'ouvrages, (en donnant la priorité aux livres français quand ils sont assez généraux).

[29] NASLIN P.

"Circuits logiques et automatismes à séquences"

DUNOD 1965

Cet ouvrage présente les méthodes traditionnelles de synthèse d'automates à séquences sous une forme appliquée. C'est un des livres qui présente le mieux le problème de la conception dans son ensemble, depuis l'énoncé du problème jusqu'à la réalisation. Il constitue ainsi un des ouvrages de base pour la conception. Il est malheureusement un peu dépassé en raison de l'évolution de la technologie.

[30] TRACEY J.H.
"Internal state assigment for asynchronous sequential machines"
IEEE Trans. EC 15, pp. 551-560, 1966

Cet article propose une des méthodes les plus célébres et les plus efficaces de codage des systèmes séquentiels asynchrones. Elle utilise la théorie des partitions [26], et un algorithme de recherche de sous matrices maximales dans une matrice booléenne. Sur ce dernier point elle s'apparente à notre méthode (chapitre 7) mais son but est uniquement de déterminer un codage minimal indépendamment du coût réel des circuits dans une technologie donnée. La méthode est plus un outil mathématique pouvant être adapté, qu'une méthode de codage directement utilisable.

[31] PERRIN J.P., DENOUETTE M., DACLIN E.
"Systèmes logiques"
DUNOD 1967.

Ce livre présente d'une façon très complète les méthodes de synthèse des systèmes logiques. Il n'est pas uniquement destiné aux ingénieurs, mais également aux chercheurs. En particulier les méthodes sont présentées sous une forme algorithmique qui les rendent facile à transposer dans une synthèse automatique.

[32] DE HAGEN

"Projet CHANCE"

Université Libre de Bruxelles, 1968.

Ce travail aborde de façon originale le problème de la synthèse des systèmes asynchrones. On trouve essentiellement deux points nouveaux :

- Un procédé de synthèse à partir d'éléments séquentiels, par identification directe des tableaux d'états (sans utiliser la notion de codage).
- Un procédé de fractionnement du tableau d'état qui donne une sorte de décomposition. Ce procédé peut permettre de retrouver la théorie de DAVID [34].

[33] NEWBORN M.M.

"A synthesis technique for binary input, binary output synchronous sequential Moore machines"

IEEE Trans., C 17, pp. 697-699, 1968

Cet article très bref jette les bases d'une méthode de synthèse utilisant des éléments exclusivement séquentiels, qui s'apparente à la méthode de DAVID [34] , bien que la théorie soit présentée dans un cas très restrictif.

[34] DAVID R.

"Réalisation de systèmes séquentiels asynchrones par interconnexion simple de cellules séquentielles identiques"

Thèse de doctorat. Grenoble 1969.

Cette thèse est le point de départ de nos travaux. Elle marque un tournant dans les méthodes de synthèse en cherchant à minimiser non pas la partie séquentielle, mais la partie combinatoire des réseaux. La méthode proposée est de beaucoup la plus simple des méthodes connues à ce jour pour la synthèse des systèmes asynchrones, et se prête bien à une conception automatique.

[35] FERRARI D., GRASSELLI A.
"A cellular structure for sequential Networks"
IEEE trans., C 18, pp.947-952. 1969

Cette étude est voisine au point de vue logique des travaux de DAVID [34] bien que l'aspect asynchrone du problème ne soit pas abordé. En outre le principe de la minimisation est plus sommaire, par exemple aucune simplification des connections n'est envisagée, l'algorithme optimise essentiellement le nombre de variables internes, et en ce sens reste assez traditionnel. La méthode de calcul du codage optimal constitue une approche différente de la nôtre (chapitre 7), et peut s'avérer intéressante si une étude complète des partitions est de toute manière envisagée.

[36] ARNOLD T.F., TAN C.J., NEWBORN M.M.
"Iteratively realized sequential circuits"
IEEE trans. C 19, pp. 54-65, 1970

Cet article envisage des méthodes de synthèses systèmatiques pour des technologies futures qui ne tiendrait pas compte du nombre de composants, mais uniquement de la structure du réseau : les exemples proposés donnent des réseaux ayant certes une structure systèmatique, mais dont le nombre d'éléments n'est pas justifié par la simplicité géométrique de la structure.

[37] BRUNO J., ALTMAN S.M.
"A theory of asynchronous control networks"
IEEE trans., C 20, pp. 629-638, 1971

Cet article qui s'inspire des travaux bien connus de LUCCONI fait ressortir l'intérêt d'une synthèse cellulaire qui copie le graphe des transitions de l'automate, en particulier dans le cas des réseaux asynchrones.

[38] MAKI G.K., TRACEY J.H.
"A state assigment procedure for asynchronous sequential circuits"
IEEE trans., C 20, pp. 666-668, 1971

Cet article utilise, pour résoudre le problème de codage des états, une méthode qui philosophiquement s'apparente à celle que nous proposons (au chapitre 7): A partir d'un codage initial plus ou moins optimisé, on construit des variables nouvelles de codage par une opération de composition qui évoque l'idée de consensus. On obtient un ensemble de variables parmi lesquelles sont choisies celles réalisant le codage minimum.

[39] TAN C.
"State assigment for asynchronous sequential machines"
IEEE trans., C 20, pp. 381-391, 1971

L'auteur présente une méthode de codage destinée, non pas à minimiser le nombre de variables internes, mais à réduire le nombre de portes des circuits combinatoires. Les résultats sont intéressants mais la complexité des circuits obtenus reste grande. Comme l'article de MAKI et TRACEY [38] l'algorithme de construction de nouvelles variables évoque l'idée de consensus.

[40] MULLER D.E., PUTZOLU G.R.
"Frequency of decomposability among machines with a large number of states"
J. Comput. & Syst. Sciences. Vol. 2, pp. 219-227, 1969
Cet article très intéressant étudie la probabilité pour qu'une machine donnée soit décomposable, au sens traditionnel d'HARTMANIS, et montre que dans certains cas cette probabilité tend vers zéro, dans d'autres elle tend vers un, quand le nombre de variables de l'automate augmente. Ce résultat fondamental permet de juger de l'opportunité de certaines méthodes, en particulier de toutes celles utilisant la théorie des partitions.

Conclusion

Cette deuxième partie de notre bibliographie fait ressortir un certain nombre de tendances nouvelles dans le domaine séquentiel.

1 - Il y a quelques années la synthèse d'un système séquentiel s'envisageait de la manière suivante : On recherchait les décompositons de l'automate, puis un codage minimum et on déterminait enfin les équations du réseau combinatoire élaborant les entrées des bascules. Cette méthode commence à être abandonnée si on en juge par la fréquence des articles ; et ceci pour deux raisons.

- La première raison concerne la théorie utilisée : décomposition et méthode de codage font intervenir la théorie des partitions de HARTMANIS, mais comme le montre [40] la méthode s'applique de plus en plus mal quand le nombre de variables d'entrées des systèmes croit.

- La deuxième raison concerne surtout le problème du codage minimum, il n'est pas du tout évident, et à notre avis faux, que le codage minimum en nombre de variables donne un coût total optimum ; d'autres paramètres peuvent intervenir : structure systèmatique dans les réseaux intégrés, coût de la partie combinatoire, temps de calcul etc...

2 - On voit apparaître une deuxième tendance, illustrée par exemple par les articles de NEWBORN [33] , DAVID [34] , FERRARI et GRASSELLI [35], ARNOLD, TAN et NEWBORN [36] .Dans ces articles on voit un soucis de traiter le problème de la conception dans son ensemble en tenant compte des contraintes de réalisations. En particulier un sujet devient à la mode, trouver une méthode de synthèse donnant un circuit de même structure que le graphe de la machine [33] et [36] .

CONCLUSION

Nous nous étions fixé comme objectif de développer des méthodes de conception entièrement automatisées des systèmes logiques. Ce but est en grande partie atteint bien que certaines méthodes ne soient pas encore programmées. Résumons brièvement ces résultats :

Nous avons étudié une première méthode de synthèse des réseaux combinatoires qui utilise une technique de calcul formel sur des expressions littérales. Cette méthode est la plus ancienne et reste relativement classique dans son principe : elle utilise les théories habituelles des formes polynominales de fonctions booléennes. Son originalité réside dans son aptitude à traiter des fonctions incomplètement spécifiées et des réseaux à sorties multiples. Des programmes trés performant ont été écrits, ils permettent de réaliser de façon économique le calcul d'un réseau combinatoire et il existe une version de ces programmes qui peut traiter des systèmes séquentiels synchrones donnés par leur tableau d'états. Les résultats fournis par ces programmes sont intéressants : les réseaux obtenus sont économiques car leur structure n'est pas forcément arborescente comme ceux obtenus par les méthodes habituelles. Enfin cette méthode est d'un emploi trés souple et peut être adaptée pour tenir compte de contraintes variées sur les aléas ou la facilité de test par exemple ; il serait intéressant et facile d'adapter les programmes à des problèmes de synthèse à l'aide d'opérateurs variés (NOR et NAND simultanément par exemple), il serait également aisé de tenir compte des contraintes de la "mise en boitier" en liant les "fan-in" des opérateurs voisins. La critique qui peut être faite à cette méthode est de n'être pas adaptée aux problèmes d'intégration à grande échelle, dans lesquels le coût des connexions intervient, c'est la raison pour laquelle nous avons étudié la méthode suivante.

La deuxième méthode qui nous proposons a été étudié pour résoudre simultanément les problèmes de synthèse, d'implantation et de câblage. Elle est beaucoup plus originale que la précédente : elle n'utilise pas la théorie des expressions polynominales, mais une notion de distance entre fonction qui combinée avec la distance géométrique des éléments logiques entre eux, nous fournit un critère d'optimalité des réseaux. Cette méthode est trés intéressante pour deux raisons : la première est d'ordre économique
- le calcul formel n'est pas simple à faire sur les ordinateurs actuels

et toutes les méthodes qui l'utilisent sont, de ce fait, assez lourdes ; par contre notre algorithme repose sur un calcul de distance qui se fait par des moyens numériques plus rapides - . La deuxième raison est d'ordre tactique - il s'est avéré plus simple au point de vue taille des programmes et temps de calcul, de traiter le problème globalement plutôt que de le décomposer en une étape de synthèse logique et une de câblage -. Actuellement cette méthode est programmée dans un cas un peu restrictif d'opérateurs NOR à deux entrées, mais pourrait être facilement étendue au cas général. Telle qu'elle est, elle a cependant fait la preuve de sa grande efficacité. L'inconvénient de mal se préter à l'élimination des aléas est largement compensé par le fait qu'actuellement cette méthode est la seule qui puisse traiter le problème du câblage sélectif dans l'intégration à trés grande échelle. Enfin rappelons que cette méthode est assez générale et a pu être utilisée pour la résolution de certains problèmes séquentiels, conjointement à la méthode qui suit.

La troisième méthode aborde le problème de la conception des systèmes séquentiels asynchrones. Elle est une généralisation de la méthode de David, et comporte plusieurs variantes. Le caractère essentiel de cette méthode est de permettre de traiter des automates trés importants et trés incomplètement spécifiés ; les programmes écrits donnent des réseaux simples dans des temps remarquablement courts, réseaux en général moins coûteux, surtout en nombre de connexions, que ceux obtenus par les méthodes habituelles. Nous avons amélioré la méthode initiale en introduisant tout d'abord une notion de consensus entre cellules, notion qui permet de réduire de manière presque optimale des automates donnés par une table d'état non primitive. Par ailleurs nous avons pu améliorer considérablement la méthode en utilisant les algorithmes précédents pour le calcul des circuits de sortie. Bien des programmes restent à écrire dans cette partie dont seule la version originale est opérationelle, mais dans leur ensemble ces méthodes sont trés nettement moins complexes que les méthodes habituelles de traitement des systèmes séquentiels asynchrones.

En définitive nos travaux contribuent à la réalisation de programmes de conception automatique, mais on peut néanmoins constater que le problème est loin d'être définitivement résolu.

(1) En premier lieu nos méthodes ne s'appliquent pas toujours avec succés : certains systèmes ayant des propriétés mathématiques bien définies sont calculés en général avec plus de facilité à la main et ne justifient pas une intervention de la machine. C'est le cas en particulier des circuits réalisant l'arithmétique d'un ordinateur ; personne ne songe à faire calculer automatiquement par exemple un circuit additionneur en binaire pur, un tel circuit existe tout fait sur le marché, depuis longtemps, et si certaines améliorations peuvent y être apportées, elles nuisent en général à à la standardisation. Nos méthodes s'appliquent plutôt aux automates dont on ne connait rien qui puisse aider le concepteur, en particulier aux automates dits industriels. Bien que la conception des ordinateurs soit plus à la mode, ce type de systèmes est celui qui nécessite le plus un traitement en machine : il existe une grande diversité de circuits, les délais de livraisons sont toujours extrêmement courts, et enfin les entreprises fabricant de tels systèmes disposent en général de personnel moins spécialisé que dans l'industrie des calculateurs.

Toutefois, il est certain qu'une adaptation des méthodes de conception automatique aux circuits d'ordinateurs serait accueillie avec intérêt par les constructeurs, et c'est vers de telles méthodes que s'orientera sans doute la recherche en conception assistée.

(2) En second lieu, il est visible que certains problèmes fondamentaux de la la conception assistée ne sont qu'esquissés à l'heure actuelle.

(a) On ne dispose pas de méthode correcte pour définir les systèmes. Certes pour la définition des ordinateurs, de nombreux langages de description ont étés développés, mais ils ne sont pas utilisables par exemple pour la desciption des automates industriels. De même certaines représentations des automates industriels ne sont pas transposables à la description d'ordinateur, ce qui rend difficile d'aborder pour ceux-ci par exemple

des problèmes tels que la décomposition. A l'heure actuelle ce problème de description est un des plus complexes, mais également celui qui donnera certainement lieu aux recherches les plus intéressantes.

(b) Le problème du test et de la maintenance des réseaux est loin d'être résolu. Si on sait (de façon assez empirique) définir la manière de tester un système quand il est déjà construit, on ne sait pas comment le construire pour que son test soit facile. Dans un de nos chapitre nous donnons une solution partielle pour le cas des systèmes combinatoires, mais rien n'est fait pour les systèmes séquentiels.

(c) Enfin il s'avère difficile en général de tenir compte des contraintes technologiques : certaines sont facilement chiffrables (nombre de composants, nombre de connexions etc....) et permettent d'élaborer un critère de minimalité des réseaux. Mais il est certain que plusieurs difficultés sont résolues plus par le "tour de main" du technologue que par une connaissance rigoureuse du phénomène. En France malheureusement il existe peu de relations entre les recherches relativement théoriques et les travaux des industriels, un gros effort reste à fournir pour combler ce fossé.

(3) Le point évoqué au paragraphe ci-dessus nous permet d'aborder un dernier aspect de la conception assistée. Actuellement si peu de constructeurs ont recours à la conception assistée, c'est probablement en raison des particularités de leurs méthodes de fabrication, de leurs critères d'évaluation du coût de leurs systèmes, de la nature des automates fabriqués. Il est évidement exclus, (même si la difficulté à définir le problème était levée), d'écrire autant de programmes qu'il y a de constructeurs distincts. Il devient essentiel de rendre les méthodes adaptables aux besoins de chacun, non pas en concevant une monstrueuse méthode universelle, mais en constituant une bibliothèque des méthodes élémentaires existantes, et en définissant un langage de macro-instructions qui permette à chacun de former sa méthode à la demande .

Le travail nécessaire pour définir ces deux outils est important et trés long comme tout travail de normalisation, mais à notre avis il conditionne absolument la diffusion réelle des méthodes de conception assistées.

TABLE DES MATIERES

		Pages
CHAPITRE I :	INTRODUCTION	2
CHAPITRE II :	THEORIE DES FONCTIONS INCOMPLETES	11
2-1.	Etude générale des fonctions incomplètes	
2-1-1.	définition et notation	12
2-1-2.	opérations	12
2-1-3.	spécifications	16
2-1-4.	forme de Lagrange	16
2-1-5.	enveloppes	17
2-1-6.	distance	18
2-1-7.	variation des fonctions	18
CHAPITRE III :	CONCEPTION DES RESEAUX COMBINATOIRES	25
3-1.	Méthode de fractionnement littérale	27
3-1-1.	opérateurs	27
3-1-2.	"Base" du fractionnement	28
3-1-3.	principe du fractionnement	28
3-1-4.	convergence	29
3-1-5.	augmentation de la "base"	33
3-1-6.	extension de la méthode	34
3-2.	Méthode topologique	36
3-2-1.	introduction	36
3-2-2.	opérateur élémentaire	37
3-2-3.	définition du fractionnement	38
3-2-4.	convergence	41
3-2-5.	choix du meilleur pivot	41

3-2-7. exemple 43
3-2-8. généralisation 44
3-3. Synthèse et implantation simultanée 44
3-3-1. structures systèmatiques des circuits 45
3-3-2. principe 46
3-3-3. remarques diverses 47

CHAPITRE IV : ALEAS
4-1. Rappel et définitions 48
4-1-2. aléas statiques et dynamiques 49
4-1-3. classification des aléas 49
4-2. correction des aléas 50
4-2-1. aléas dans un fractionnement 50
4-2-2. définition d'un fractionnement correct 51
4-2-3. aléas dynamiques 54

CHAPITRE V : TEST DES SYSTEMES COMBINATOIRES 58
5-1. considérations technologiques 59
5-2. application à la synthèse par fractionnement 59
5-3. algorithmes de fractionnements 62
5-4. généralisations à des opérateurs quelconques et fonctions incomplètes 68

CHAPITRE VI : SYSTEMES ASYNCHRONES, DEFINITIONS 70
6-1. représentation 71
6-2. définitions diverses 71
6-3. relations 72
6-4. relations d'hypovalences, applications 74
6-4-1. Cellules de DAVID 75
6-4-2. utilisation de la cellule 75
6-4-3. théorème fondamental 77

6-4-4. Etats sources 79

6-4-5. Suppression d'entrées primaires 82

6-4-6. Groupement d'états 84

CHAPITRE VII : THEORIE DU CONSENSUS ENTRE ETATS 85

7-1. Introduction à l'idée de consensus 86

7-2. définitions : descendants, descendants nuls et non spécifiés 88

7-3. Relations entre variables 92

7-4. Treillis des lignes 94

7-5. Consensus 96

7-6. Simplifications 98

7-7. Méthode générale de minimisations 101

7-8. Exemple 103

CHAPITRE VIII : ETUDE DES SORTIES 108

8-1. Remarques : utilisation des NOR, existences de relations 108

8-2. Détermination de la base initiale 110

8-2-1. Hypovalence virtuelle 110

8-2-2. Algorithme de recherche d'une base 111

8-2-3. Exemple 111

8-2-4. Théorème 113

8-2-5. Choix des relations virtuelles 113

8-3. Méthode algébrique 116

8-4. Méthode topologique 117

CHAPITRE IX : DECOMPOSITION DES AUTOMATES 121

9-1. Décomposition en somme et produit, définition 122

9-2. Exemple 123

9-3. Etude mathématique, décomposition d'après une relation. 125

CHAPITRE X : BIBLIOGRAPHIE

CONCLUSION

VU

Grenoble, le

Le Président de la Thèse

VU

Grenoble, le

Le Doyen de la Faculté des Sciences

VU, et permis d'imprimer
Le Recteur de l'Académie de GRENOBLE