Zeitschrift: Bulletin des Schweizerischen Elektrotechnischen Vereins, des Verbandes Schweizerischer Elektrizitätsunternehmen = Bulletin de l'Association suisse des électriciens, de l'Association des entreprises électriques suisses

Herausgeber: Schweizerischer Elektrotechnischer Verein ; Verband Schweizerischer Elektrizitätsunternehmen

Band: 79 (1988)

Heft: 11

Artikel: De la machine séquentielle synchronisée au processeur Risc

Autor: Piguet, C. / Pache, R. / Aubert, P.

DOI: https://doi.org/10.5169/seals-904043

De la machine séquentielle synchronisée au processeur Risc

C. Piguet, R. Pache, Ph. Aubert, P. Clément, B. Perrin, JF Perotto, M. Ansorge, JF Joss, H. Leuthold

Les circuits intégrés à applications spécifiques (ASIC) sont généralement constitués d'une unité de traitement et d'une unité de commande. La méthode de conception d'unités de traitement est bien connue aujourd'hui. Par contre, les méthodes de conception des unités de commande sont fort diverses et produisent des résultats fort dissemblables. Cet article présente toute une série de circuits intégrés industriels, réalisés au CSEM, dont l'unité de commande a été réalisée sous forme programmée, utilisant les concepts des machines de décision binaire.

Anwendungsspezifische integrierte Schaltungen bestehen grundsätzlich aus einer Verarbeitungseinheit und einer Befehlseinheit. Während der Entwurf von Verarbeitungseinheiten keine Probleme bietet, liefern die verschiedenen Entwurfsmethoden für Befehlseinheiten noch immer sehr unterschiedliche Resultate. Der vorliegende Beitrag stellt einige im CSEM realisierte ICs vor, deren Befehlseinheit unter Zuhilfenahme der Methode der Binärentscheidungs-Maschine konzipiert wurde.

Adresse des auteurs

C. Piguet, R. Pache, Ph. Aubert, P. Clément, B. Perrin, JF Perotto, M. Ansorge, JF Joss, H. Leuthold, Centre Suisse d'Electronique et de Microtechnique (CSEM), rue de la Maladière 71, 2000 Neuchâtel.

La description d'une architecture interne d'un microprocesseur peut s'effectuer en décrivant chacun de ses éléments logiques les uns après les autres, puis à supposer que le lecteur devinera comment ils fonctionnent tous ensembles. Cet article propose une autre démarche: celle de partir d'une machine logique très simple (la machine séquentielle synchronisée ou MSS des cours de Systèmes logiques) pour en déduire, après des transformations, le fonctionnement interne d'un microprocesseur. Celui-ci sera très simple, comportant peu d'instructions, et il est intéressant de constater qu'il sera classé comme Risc (Reduced Instruction Set Computer). Or, les machines Risc sont un sujet de recherche très actuel.

La démarche proposée passe par l'analyse des machines de décision binaires, qui sont le fondement du passage du matériel au logiciel. Une même fonction logique peut être réalisée matériellement par des portes logiques ou au contraire être réalisée par un programme exécuté par une machine de décision binaire. Or, le fonctionnement d'un processeur peut être compris en se basant sur celui des machines de décision binaire.

Un autre but de cet article est de montrer que toutes ces machines ont des applications industrielles, en particulier dans la conception de circuits intégrés. Ainsi, cet article propose pour chaque type de machine une réalisation industrielle. Ces descriptions sont relativement détaillées et, bien qu'intéressantes, elles ne sont pas essentielles à la compréhension de la démarche proposée.

Situation actuelle

Les circuits à applications spécifiques (ASIC) peuvent être décomposés en une *unité de traitement* et une *unité de commande*. L'unité de traitement permet de réaliser des opérations arithmétiques et logiques sur des données binaires. L'unité de commande permet de commander la séquence de ces opérations.

Des outils de Conception Assistée par Ordinateur (CAO) permettent aujourd'hui de créer facilement les plans de masques ou layouts de tels circuits. La technique des Cellules standards [1] est bien connue, et les compilateurs de silicium seront les outils du futur. C'est le but de cet article de montrer que si les outils CAO sont nécessaires, ils ne sont pas suffisants au développement de circuits intégrés performants. Nous pensons en effet que les choix architecturaux sont plus importants pour obtenir rapidement un circuit performant.

Les architectures et les méthodes de conception des unités de traitement sont bien connues. Comme ces unités travaillent sur des mots de n bits, elles sont naturellement décomposées en n tranches qui sont assemblées pour réaliser l'unité complète. L'unité de traitement est par conséquent très régulière, et si elle occupe généralement la majeure partie de la surface de la puce, elle ne demande généralement qu'un temps de développement relativement court.

Les méthodes de conception des unités de commande ne sont pas parvenues à un tel degré de maturité. Généralement, le temps de développement de telles unités est assez long, bien qu'elles n'occupent qu'une faible partie de la surface de la puce. Toute une série d'architectures sont possibles. Il y a quelques années, les unités de commande de microprocesseurs étaient réalisées en logique câblée. Aujourd'hui, les unités de commande des microprocesseurs complexes de 16 ou 32 bit sont microprogrammées [2; 3]. Cette évolution est également apparente pour les circuits à applications spécifiques. C'est également un but de

cet article de montrer que les unités de commande microprogrammées peuvent être utilisées avec profit pour des circuits à applications spécifiques.

Les ingénieurs-systèmes dans l'industrie sont généralement habitués à utiliser de la logique câblée. Il est par conséquent très simple, dans la réalisation d'un circuit intégré, d'avoir recours à la méthode des Cellules standards pour traduire en layout le schéma logique du circuit, constitué de portes et de bascules. Mais nous pensons qu'une unité de commande programmée est encore plus simple à concevoir. Cet article présentera des exemples d'unités de commande programmées, réalisées au CSEM pour des circuits intégrés industriels.

L'utilisation industrielle d'unités de commande programmées illustre également une excellente collaboration qui s'est instaurée entre le CSEM et le Laboratoire de Systèmes logiques (LSL) de l'Ecole Polytechnique Fédérale de Lausanne. Quantité d'idées sont nées au LSL, en particulier de la part du Prof. *D. Mange* et du Dr *E. Sanchez*, et ont été appliquées aux circuits industriels présentés dans cet article.

Unités de commande câblées ou programmées

1. Machines séquentielles synchronisées

La commande d'une unité de traitement peut être effectuée par une machine séquentielle synchronisée [4] ou MSS. Une MSS admet comme mode de représentation un graphe des états ou une table des états. Un signal d'horloge est utilisé pour synchroniser la machine. Pour une machine dite de *Moore*, les sorties sont fonction de l'état interne mémorisé dans un registre, tandis que pour une machine dite de *Mealy*, les sorties sont fonction à la fois de l'état interne et des entrées.

La structure de base d'une MSS est représentée à la figure 1. Elle est composée d'un circuit logique combinatoire et d'un registre synchrone. Les sorties du registre mémorisant l'état interne de la machine constituent des entrées du circuit logique combinatoire. Celui-ci est réalisé par des portes logiques pour une version câblée ou par une mémoire ROM (Read Only Memory) pour une version programmée.

Une MSS dont le circuit logique combinatoire est réalisé en logique câblée exige du concepteur la simplification des équations logiques. Si cette simplification est faite à la main, il n'est possible de réaliser que des MSS très simples. L'utilisation d'un PLA (Programmed Logic Array) revient au même sur le plan de la méthode: un PLA exige aussi que les équations logiques soient simplifiées. Un PLA est donc seulement une manière d'organiser un ensemble de portes logiques. Il existe de nombreux programmes CAO pour obtenir les équations logiques simplifiées d'un PLA.

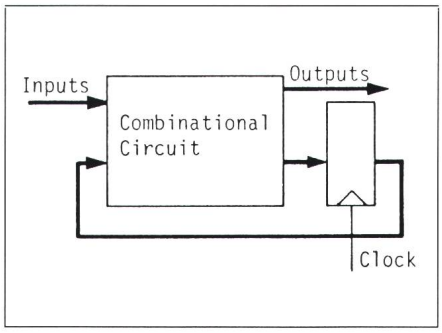


Figure 1 Machine séquentielle synchronisée (MSS)

2. Machines séquentielles synchronisées programmables

Une MSS programmable [5] comporte une mémoire ROM pour la réalisation du circuit logique combinatoire. Une telle réalisation n'exige pas la simplification des équations logiques. La table d'états peut être directement programmée dans la mémoire ROM. Si la conception est faite à la main, cela peut être un avantage important. Le prix à payer est une surface de silicium plus importante que le PLA correspondant, mais pour de petites MSS, ce point n'est pas essentiel.

Un avantage plus déterminant est le fait qu'une MSS programmable peut être facilement modifiée. Si l'on suppose que le nombre des entrées et des sorties (binaires) ne change pas, ainsi que le nombre de bits du registre, une erreur dans la table d'états ou une modification des spécifications peut être aisément reprogrammée dans la mémoire ROM. La forme du layout de la machine, ainsi que les positions des entrées et des sorties, ne sont pas affectés par cette reprogrammation. Cela est loin d'être le cas pour une MSS dont le circuit combinatoire est réalisé à l'aide de portes ou d'un PLA, tout simplement parce qu'une autre simplification logique donnera des équations plus simples ou plus complexes. Même si le layout est produit automatiquement, il sera différent de la première version, et tout le routage devra être refait.

3. MSS programmable industrielle

La figure 2 représente le plan directeur d'une machine séquentielle synchronisée programmable et industrielle qui a été incorporée comme périphérique à un processeur horloger, qui est décrit dans ce même article (fig. 15). Cette machine a pour rôle la gestion des entrées d'une montre [6], réalisées sous forme de quatre touches disposées les unes à côté des autres sur le bas du cadran. L'utilisateur peut effleurer ces touches dans un sens ou dans l'autre pour indiquer qu'il aimerait avancer ou reculer l'heure de sa montre. La machine décrite doit détecter le sens du mouvement du doigt de l'utilisateur, en analysant la séquence des codes qui sont fournis par les touches. Un timer indique encore à cette machine que l'utilisateur n'a pas pressé de touche depuis assez longtemps. Les sorties de cette machine indiquent au processeur horloger ce qu'il doit faire, à savoir incrémenter ou décrémenter les secondes, les minutes ou les heures, selon le sens du mouvement du doigt de l'utilisateur.

Les spécifications de cette machine peuvent être formalisées en un graphe de 14 états. Le registre synchrone contient alors 4 bit qui suffisent pour coder les états internes. La table d'états correspondante a été directement programmée dans la mémoire ROM de cette machine.

Les entrées de cette machine, à savoir les quatre touches et la sortie du timer (fig. 2), peuvent être codées sur trois bits par un transcodeur, tout simplement parce qu'il n'existe que 8 configurations des cinq entrées sur les 32 possibles. Les sorties de ce transcodeur sont stockées dans un registre d'entrée de 3 bit, nommés *X*, *Y* et *Z* sur la figure 2.

La figure 3 représente le layout correspondant au plan directeur de la figure 2. La mémoire ROM est réalisée par une logique à précharge [7]. Le transcodeur ainsi que le registre synchrone sont réalisés en layout orienté [8]. Les connexions entre le registre et la ROM passent au travers du décodeur d'adresses de la ROM. Cela implique que le tableau ainsi que le décodeur d'adresses de la ROM ont été un

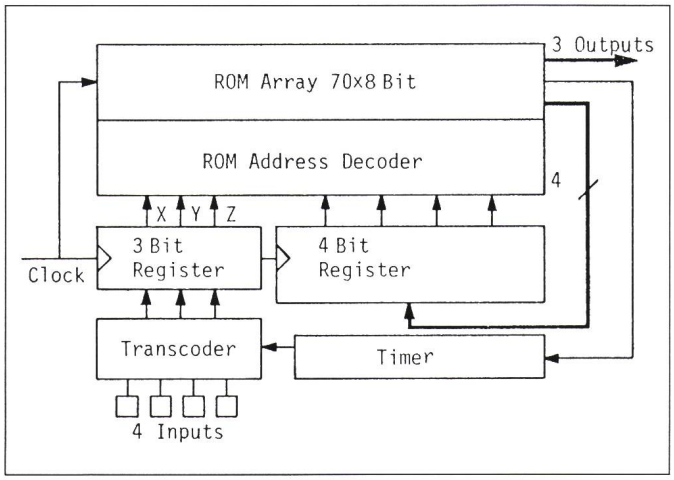


Figure 2 Plan directeur d'une MSS programmable

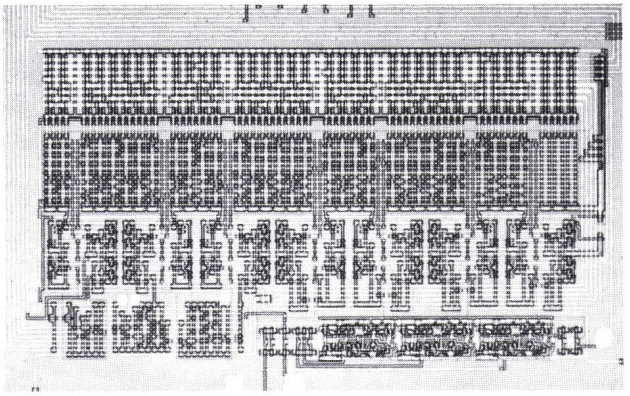


Figure 3 Microphotographie de la MSS programmable

peu étendus pour laisser la place à ces connexions.

Ce circuit comporte environ 1000 transistors, en comptant une mémoire ROM non complète de 70 mots de 8 bit. Un plan directeur a été dressé avant la conception du registre, dont la longueur a été choisie de manière à s'adapter à celle de la ROM. Ce circuit présente une excellente densité d'intégration, due bien évidemment en partie à la densité de la ROM, et un temps de développement très court. La première intégration était fonctionnelle.

Machines de décision binaire

1. Principe

Les entrées des machines séquentielles synchronisées programmables présentées à la section précédente étaient directement connectées au décodeur d'adresses de la mémoire ROM. Toutes les entrées sont par conséquent testées simultanément pour déterminer l'état suivant. Si l'on suppose une machine avec n entrées et un registre de m bits, la mémoire ROM contient alors 2^{n+m} mots. Pour de grandes valeurs de n et m, la mémoire ROM peut alors atteindre une taille prohibitive.

Les machines de décision binaire [9; 10; 11] permettent de réduire la taille de la mémoire ROM si l'on accepte d'exécuter plusieurs étapes pour déterminer l'état futur. L'idée de base est de considérer séquentiellement les n entrées. Les machines de décision binaire ont par conséquent besoin de davantage de coups d'horloge que les MSS pour exécuter une tâche. Une instruction de test permet de sélectionner une des n entrées et d'effectuer un branchement à une adresse spécifiée dépendante de la valeur de la variable d'entrée sélectionnée.

La figure 4 représente une architecture possible d'une machine de décision binaire. La mémoire ROM est adressée par un registre. Un champ de l'instruction de test permet de choisir une variable d'entrée par un multiplexeur de test. En commandant un multiplexeur d'adresses, la sortie du multiplexeur de test sélectionne un des deux champs d'adresse contenu dans l'instruction de test. La taille de la ROM n'est ainsi plus dépendante du nombre d'entrées.

2. Algorithmes

On peut montrer que toute fonction logique peut être représentée par un algorithme de décision binaire [9; 12]. La figure 5a représente une fonction logique donnée par une table de Karnaugh. En appliquant une méthode de simplification qui partage successivement la table de Karnaugh (fig. 5b), il est facile d'obtenir un algorithme de décision binaire (fig. 5c). Les blocs de zéros et de uns doivent couvrir toute la table sans avoir de recouvrements [12]. Les algorithmes de décision binaire contiennent deux types d'instructions, à savoir une instruction de test (if...then...else) et une instruction de sortie (do...). Cette dernière est utilisée pour charger une valeur dans un registre de sortie. Un algorithme de décision binaire peut être écrit dans un langage utilisant les instructions if...then...else et do... (fig. 5c).

L'exemple de la figure 5 permet de montrer que toute fonction logique peut être réalisée par un algorithme de décision binaire. Pour obtenir un résultat en sortie, il est néanmoins nécessaire d'exécuter plusieurs instructions.

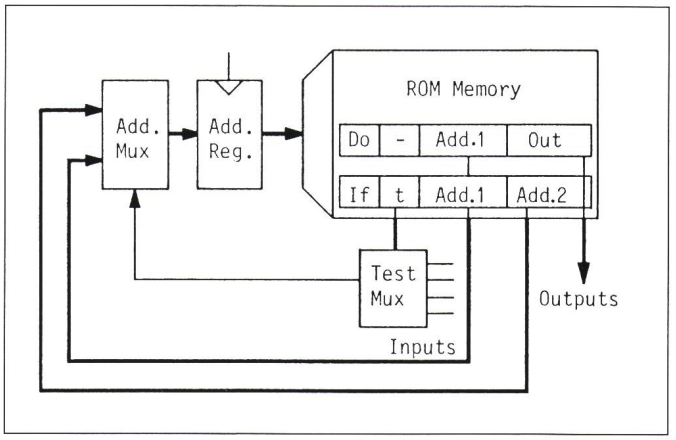


Figure 4 Architecture d'une machine de décision binaire

Add. Mux
Multiplexeur d'adresses

Add. Reg.
Registre d'adresses

La sortie n'est donc disponible qu'après plusieurs coups d'horloge.

3. Tables d'états réalisées

Une table d'état peut être simplifiée selon la méthode décrite à la section précédente pour déterminer son algorithme de décision binaire. Nous prenons en exemple la table d'états correspondant à la MSS décrite à la figure 2 et nous cherchons à l'implémenter par une machine de décision binaire. Une méthode possible est d'utiliser chaque état du graphe d'états pour déterminer pour chacun d'entre eux un algorithme de décision binaire qui donne l'adresse de l'état futur [13]. La figure 6 représente un des 14 états de cette MSS. L'état futur dépend des trois variables d'entrée *X*, *Y* et *Z* (fig. 2). Une table de Karnaugh peut être obtenue à partir de ce graphe partiel pour calculer l'algorithme de décision binaire. Pour cet état, l'algorithme comporte 4 instructions (fig. 6). Pour le graphe des états complet, formé des 14 algorithmes de décision binaire correspondant aux 14 états du graphe, le nombre total d'instructions peut être estimé à 14×4 instructions, soit environ 56 instructions. Cela implique un registre d'état de 6 bit au lieu du registre de 4 bit de la MSS (fig. 2). Les instructions comportent un bit de code opératoire, 2 bit pour sélectionner une des trois entrées et deux champs d'adresses de 6 bit chacun. L'instruction comporte ainsi 15 bit. Dans ce cas particulier, à la fois les tailles de la ROM et de son registre d'adresses sont plus grandes que celles de la MSS.

Ainsi, et contrairement à l'affirmation à la section 1 de ce chapitre, au lieu de diminuer la taille de la ROM, le recours à une machine de décision binaire pour de petites machines peut avoir pour résultat d'augmenter la taille de la ROM. Cela explique pourquoi une machine de décision binaire n'est pas souvent utilisée pour implémenter une simple table d'états. Elle est seulement intéressante dans le cas où, à partir d'un état, il n'existe que peu d'états futurs, comme dans les compteurs [13].

Il est néanmoins certain que seules des tâches simples peuvent être formalisées à l'aide de tables d'états ou de graphes d'états. Pour des tâches complexes, il est nécessaire de recourir à d'autres moyens de représentation, comme un programme. Dans ce cas, il est possible d'écrire directement le programme d'une machine de décision binaire. La méthode de simplification décrite à la section précédente reste utilisable pour de petites procédures de ce programme.

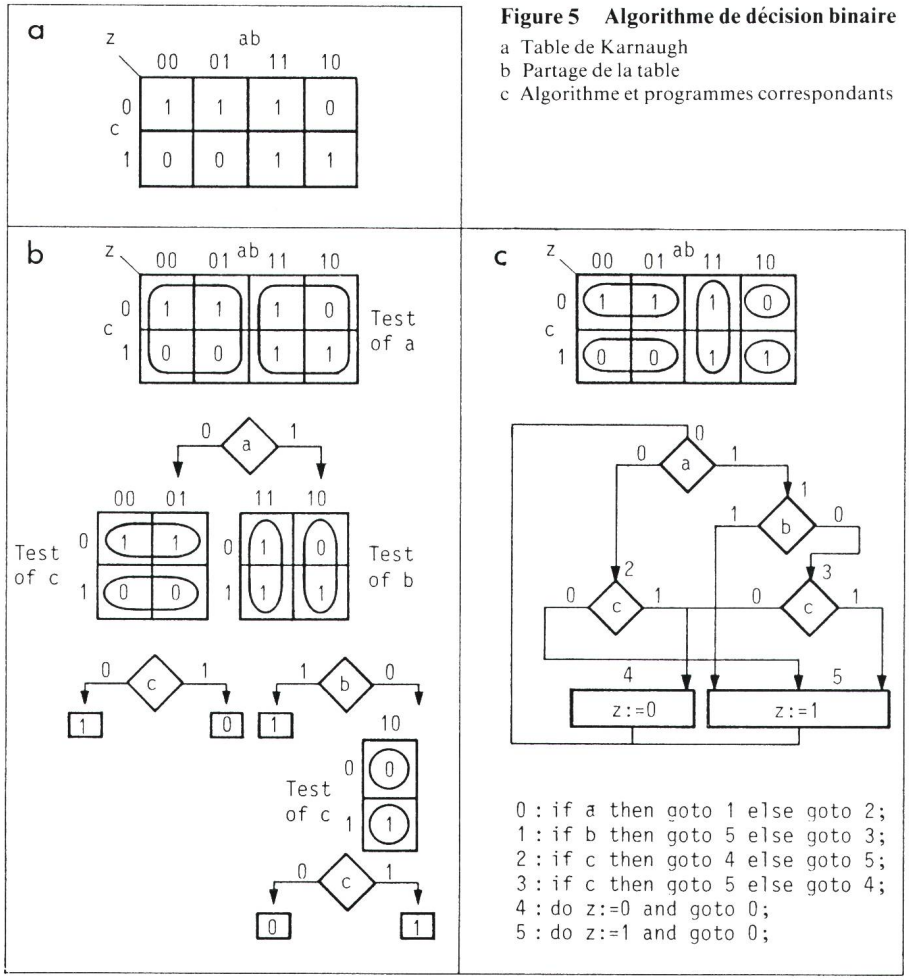


Figure 5 Algorithme de décision binaire
a Table de Karnaugh
b Partage de la table
c Algorithme et programmes correspondants

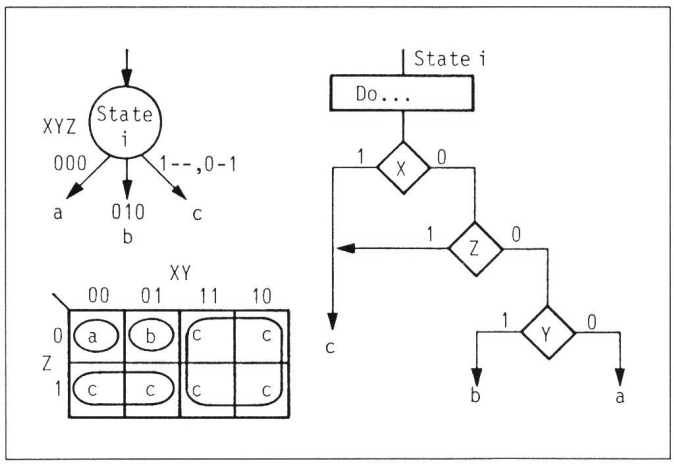
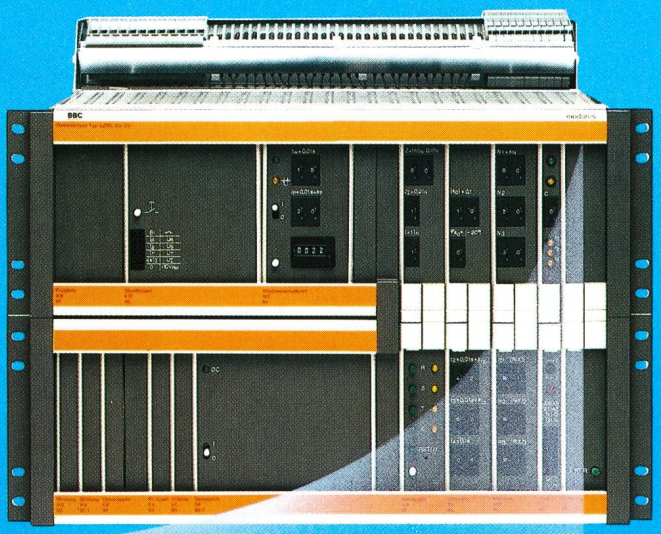


Figure 6 Algorithme de décision binaire pour un graphe des états.
Les traits dans l'expression (1--, 0-1) indiquent que C est indépendant de la variable ainsi marquée.

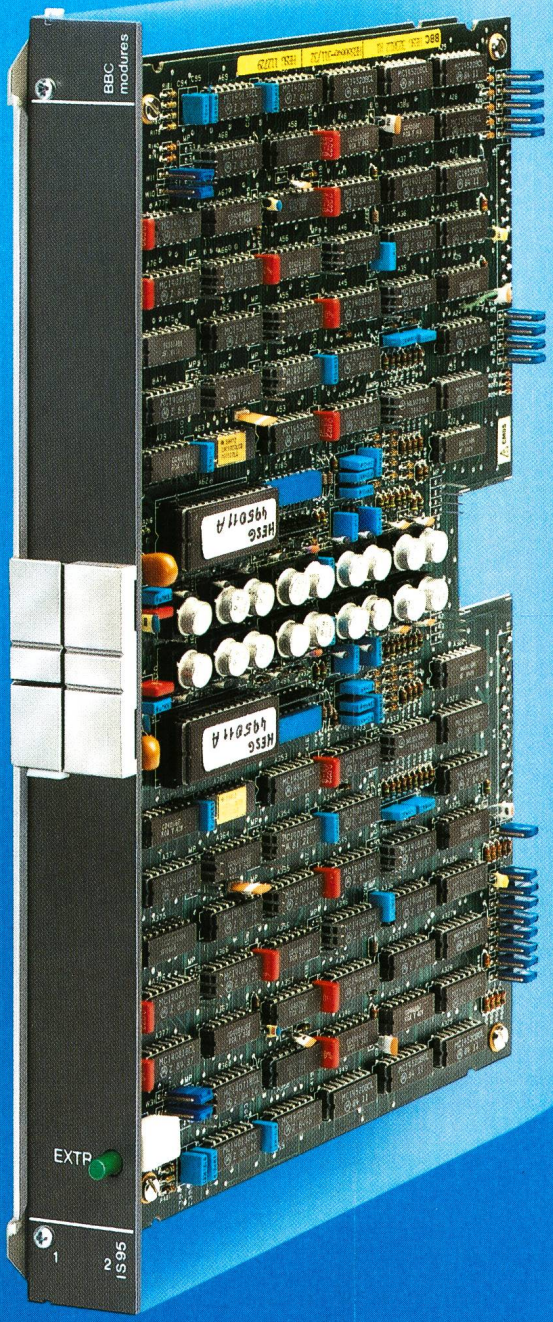
4. Machines de décision binaire pour des tâches complexes

La machine de décision binaire représentée à la figure 4 n'est pas bien adaptée à l'exécution de tâches complexes. Si plus de 1000 instructions sont nécessaires, les adresses doivent être codées sur 10 ou 11 bit. Il est alors plus économique de recourir à un compteur de programme plutôt que d'avoir une instruction de test avec deux champs d'adresses de 10 ou 11 bit. Ainsi, l'instruction de test devient

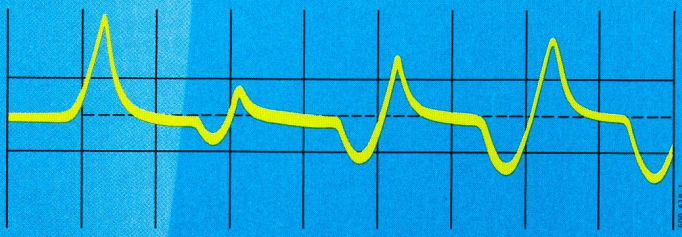
Asea Brown Boveri macht den Distanzschutz noch sicherer.



Das neuartige Distanzrelais Typ LZ95 misst immer richtig, auch bei gesättigtem Stromwandler!

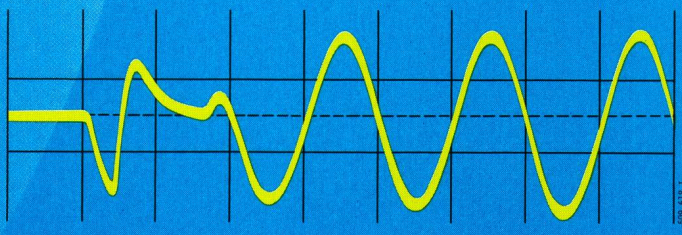


Sollte das Ausgangssignal Ihres Stromwandlers infolge Sättigung im Kurzschlussfall einmal so aussehen:



so ist dies nicht weiter schlimm, sofern ein Distanzrelais Typ LZ95 Ihre Anlage schützt.

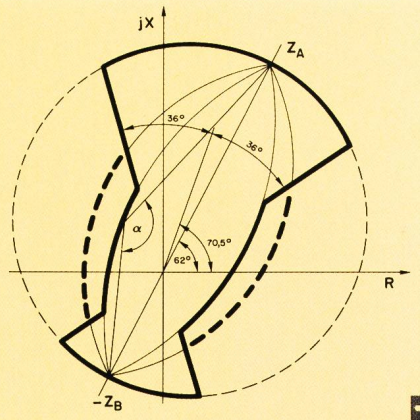
Dieses Relais ist intelligent genug, aus den ungesättigten Fragmenten den wahren Kurzschlussstrom nachzubilden.



Deshalb misst und reagiert dieses Relais immer richtig!

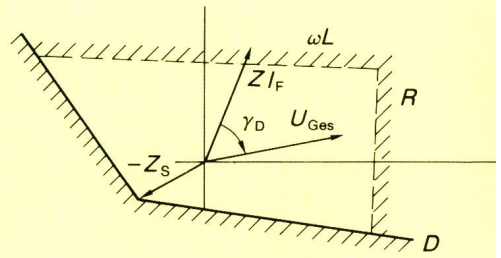
ABB
ASEA BROWN BOVERI

Eigenschaften und Anwendungsgebiete

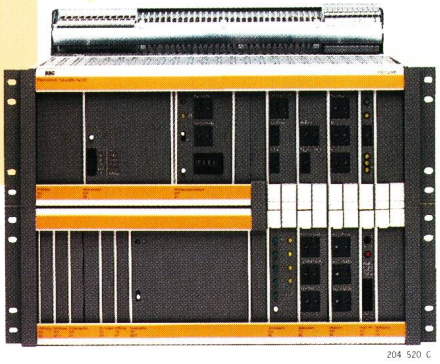


Optimale Kennlinie der Unterimpedanzanregung

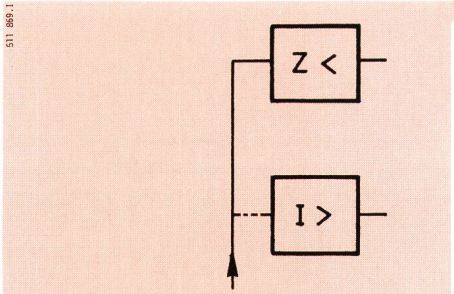
Interne Überwachung
Diagnose
Frontmessbuchsen



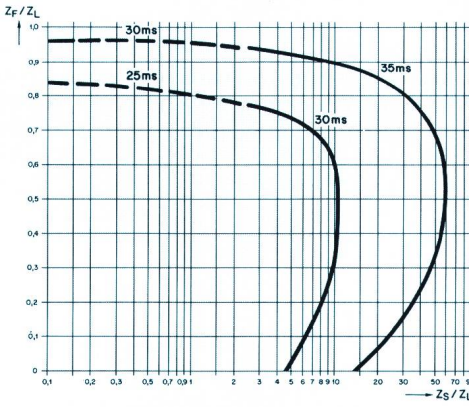
Polygonale Kennlinie mit unbegrenzter Richtungsempfindlichkeit
Einstellungen getrennt pro Stufe



Alle üblichen HF-Arten



Unterimpedanz- und Überstromanregung (für Nahfehler mit höherem Strom), getrennt blockierbar, z.B. durch das neu konzipierte System für die Überwachung der Spannungswandlerkreise (UV 95)



Kurze Auslösezeiten

Typische Einsatzgebiete:

- Kabel, Freileitungen oder hoch belastete lange Leitungen
- Übertragungs- oder Verteilnetze, unabhängig von der Sternpunktbehandlung
- Induktive oder kapazitive Spannungswandler
- Anlagen mit schwachen Hauptstromwandlern infolge von gestiegenen Kurzschlussströmen oder wegen Platzmangels (z.B. in SF_6-Anlagen).

Ich wünsche:

- ☐ weitere Unterlagen über Schutztechnik
- ☐ Telefonkontakt
- ☐ Besuch

Name und Vorname

Strasse/Nr.

PLZ/Ort

Telefon-Nr.

Firma

Unterschrift

Nicht frankieren
Ne pas affranchir
Non affrancare

Geschäftsantwortsendung **Invio commerciale-risposta**
Correspondance commerciale-réponse

CH-IT 5084 87 D

Asea Brown Boveri AG
Verkauf Schweiz

5401 Baden

ASEA BROWN BOVERI

Asea Brown Boveri AG
Verkauf Schweiz
5401 Baden

un branchement conditionnel du type if...then branch to... Si le branchement n'est pas effectué, le compteur de programme est incrémenté. Les algorithmes de décision binaire peuvent être programmés à l'aide de ce type d'instructions. Les instructions de branchement non conditionnels sont alors réalisées en testant une entrée particulière qui est toujours câblée au «1» logique.

Pour des tâches complexes, il est généralement utile de pouvoir écrire des sous-programmes. Deux instructions supplémentaires sont alors nécessaires, à savoir une instruction d'appel (call) et de retour de sous-programme (return). L'adresse de retour doit être mémorisée dans une pile. La figure 7 représente une machine de décision binaire comportant un ensemble de compteurs de programme. Ceux-ci sont constitués d'un maître M et d'un ensemble d'esclaves PC. Un appel à un sous-programme est réalisé en changeant simplement de PC, ceux-ci étant adressés par un pointeur de pile SP (compteur-décompteur). Le PC utilisé avant l'appel conserve naturellement l'adresse de retour sans qu'il soit besoin d'exécuter la moindre procédure de sauvetage. Il a été montré [14] que cette architecture est très intéressante pour une réalisation VLSI, du fait de sa régularité et de sa facilité de connexion avec la mémoire ROM.

De telles machines de décision binaire empruntent plusieurs caractéristiques aux microprocesseurs, en particulier la présence de compteurs de programme et des instructions d'appel et de retour de sous-programmes. Elles sont programmées de la même manière que le sont les microprocesseurs. Néanmoins, il existe des différences importantes. Le répertoire d'instructions est limité à quatre types d'instructions. Il est bien évident que l'instruction de sortie «do...» est utilisée pour commander différentes opérations de l'unité de traitement. On prend donc l'habitude de décrire autant d'instructions «do...» différentes qu'il y a d'opérations effectuées par l'unité de traitement. Il en résulte que le répertoire d'instructions comporte alors 10 à 20 instructions. Une autre différence importante par rapport aux microprocesseurs est que l'instruction est codée sur un seul mot. La longueur du mot dépend de l'application, mais compte certainement plus de bit que le célèbre 8 bit des microprocesseurs. Chaque instruction est exécutée en un seul cycle de plusieurs phases, mais jamais, comme les microprocesseurs, en un nombre variable de cycles.

Figure 7
Architecture d'une machine de décision binaire avec compteur de programme et pile

PC Program Counter
SP Stack Pointer
M Master PC
INC Incrementer
IR Instruction Register

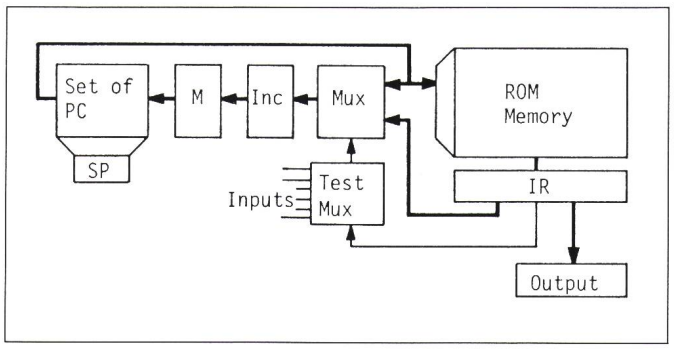
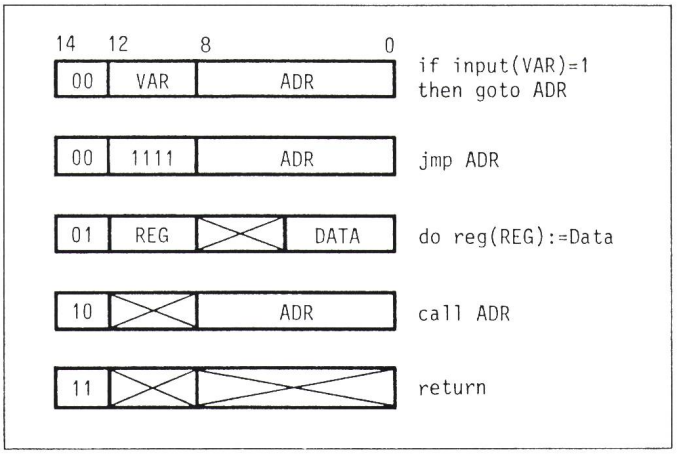


Figure 8
Répertoire d'instructions



5. Machine de décision binaire industrielle

Les machines de décision binaire ne sont pas souvent utilisées pour des circuits intégrés industriels. La machine décrite ici (fig. 8 et 9) a été conçue pour un circuit à application médicale. L'unité de traitement était principalement composée de compteurs et de périphériques d'entrée et de sortie.

Le répertoire d'instructions de cette machine est représenté à la figure 8. Un code opératoire de 2 bit permet de distinguer quatre types d'instructions, à savoir le branchement, le «do», le «call» et le «return». L'instruction de

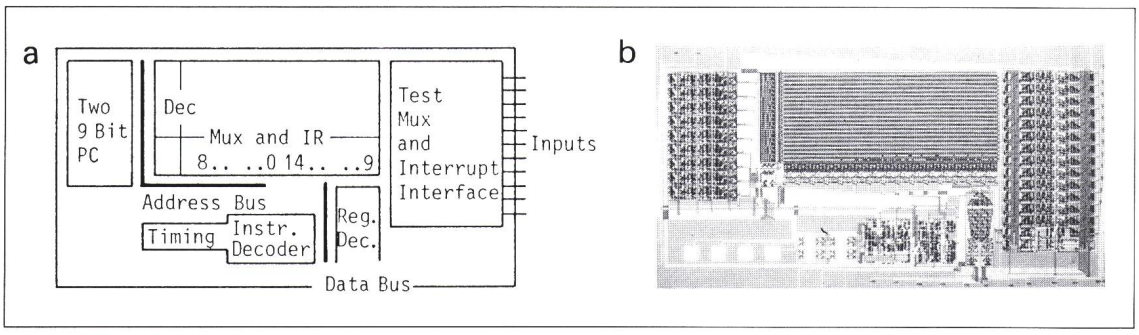


Figure 9
Machine de décision binaire industrielle

a Plan directeur
Reg. Dec. Décodeur d'adresses des registres de sortie

b Microphotographie

branchement conditionnel permet de tester 16 signaux d'entrée. Une de ces entrées (adressée par 1111) est câblée au «1» logique, ce qui permet de définir la 5e instruction du répertoire, qui est le branchement non conditionnel. L'adresse de branchement est codée sur 9 bit, ce qui permet d'adresser 512 instructions. L'instruction «call» comporte ainsi un champ d'adresse de 9 bit et l'instruction «return» n'a pas de paramètre. L'instruction «do» comporte deux paramètres, à savoir l'adresse d'un des 16 registres de sortie et la donnée de 4 bit à charger. L'instruction comporte 15 bit et la mémoire ROM 7680 bit.[1]

L'architecture de la machine de décision binaire est très semblable à celle de la figure 7. La pile des compteurs de programme ne comporte que deux niveaux, si bien que le pointeur de pile n'est qu'une simple bascule, pointant tantôt le premier compteur de programme, tantôt le deuxième. D'autres circuits sont nécessaires pour cette application particulière, comme le décodeur d'adresses des registres de sortie ou l'interface pour des interruptions. La figure 9a représente le plan directeur de cette machine, tandis que la figure 9b représente le circuit intégré test correspondant. Celui-ci a été réalisé avant l'intégration du circuit complet [15]. Pour des raisons topologiques, l'ordre des bit de l'instruction a été modifié sur le layout par rapport au répertoire d'instructions. Les parties en logique câblée ont été réalisées en layout ordonné [8].

Figure 10
Architecture d'une machine à 8-décision
M1, M2, M3: champs de sélection d'une variable d'entrée
A1, ..., A8 champs d'adresse

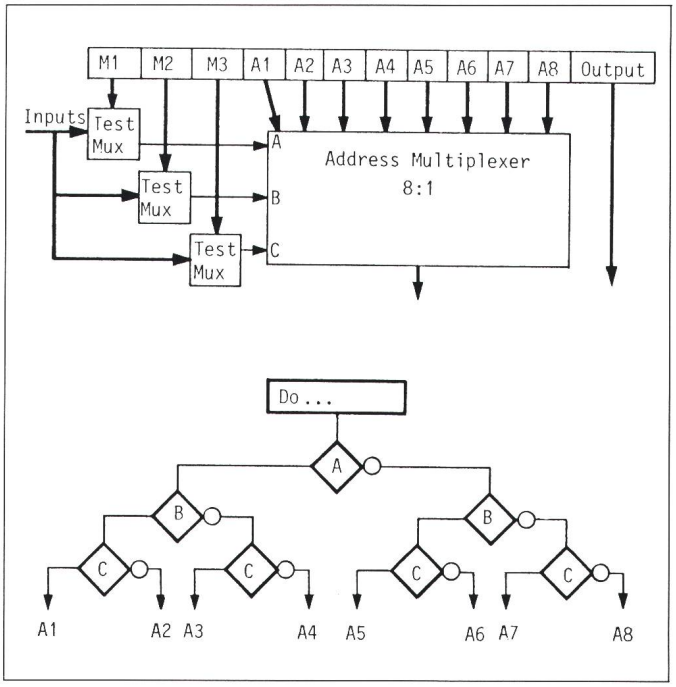
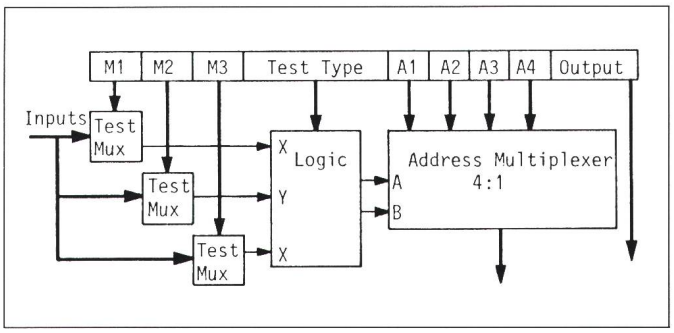


Figure 11
Architecture d'une machine à 4-décision avec prétraitement des entrées



Machines à N-décision

1. Principe des machines à N-décision

La machine de décision binaire (ou 2-décision) est basée sur un test séquentiel des variables d'entrée. Pour certaines applications, il n'est pas acceptable d'exécuter plusieurs instructions pour déterminer l'état futur de la MSS équivalente (fig. 6). Il a été démontré que pour certains problèmes [16], une machine de N-décision (où $2 < N < 2^n$, s'il y a n entrées) peut atteindre la même vitesse qu'une MSS, mais avec un coût réduit.

L'idée de base est que certaines instructions puissent tester simultanément plusieurs variables d'entrée au lieu d'une seule pour la machine de décision binaire. La figure 10 représente une machine à N-décision [16; 17] où plusieurs multiplexeurs de test permettent de tester plusieurs variables à la fois. Toutes les variables d'entrée sont connectées à chaque multiplexeur. Ces multiplexeurs sont commandés chacun par un champ différent de l'instruction de test. Les sorties des multiplexeurs de test sont connectées aux entrées de sélection du multiplexeur d'adresses. Celui-ci est capable de choisir une adresse parmi $N=2^k$ adresses, s'il y a k multiplexeurs de test et k variables d'entrées testées simultanément. La figure 10 représente une machine à 8-décision capable de tester simultanément trois variables d'entrée. Cette machine exécute en un seul coup d'horloge l'instruction de test multibranchements de la figure 10.

2. Machines à N-décision avec prétraitement des entrées

Il est évident que pour l'instruction de la figure 6, une machine à 8-décision est loin d'être optimale. Il est possible d'ajouter un circuit logique entre les multiplexeurs de test et le multiplexeur d'adresses, comme le représente la figure 11. Cette architecture est une machine à 4-décision, qui peut néanmoins tester 3 variables d'entrée simultanément. Elle est donc mieux adaptée à l'exécution de l'instruction de la figure 6. Le circuit logique additionnel réalise un prétraitement des entrées [17]. L'idée de base est de définir plusieurs instructions de test différentes, chacune d'entre elles permet-

[1] Cette machine de décision binaire comporte environ 9400 transistors, y compris la ROM de 7680 bit. La régularité [14] d'environ 15,3, incluant la ROM, est tout à fait acceptable. Les 1700 transistors de la partie en logique câblée présentent une régularité de seulement 3,3. Le temps de conception fut considéré comme très intéressant par rapport à ceux de circuits équivalents entièrement en logique câblée.

tant de tester une seule configuration particulière des variables d'entrée. Le choix de ces instructions de test est effectué en analysant soigneusement le programme à réaliser.

3. Machine à 4-décision industrielle

Une machine à 4-décision a été réalisée (fig. 11) pour la commande d'un convertisseur analogique-numérique basé sur des approximations successives. Chaque bit doit être converti sériellement à 10 kHz. Trois signaux d'entrée, à savoir SP (début de conversion), OR (interruption) et AC (redémarrage d'une conversion), doivent être pris en compte en 100 µs.

L'analyse du fonctionnement (fig. 12) a montré que les quatre instructions suivantes, à savoir le test de SP, la commande Start A/D, le test de OR et le test de AC, doivent être exécutées en 100 µs. Si l'on utilise une machine de décision binaire où ces 4 instructions sont exécutées en séquence, une horloge de 40 kHz est nécessaire. Par contre, si l'on utilise une machine à 4-décision, permettant de tester simultanément ces trois variables d'entrées et d'envoyer la commande en une seule instruction, la fréquence d'horloge est alors de 10 kHz, ce qui réduit la consommation.

Différents types d'instructions de test sont nécessaires, le premier (instr. 0) avec quatre adresses futures, le deuxième avec trois et le dernier avec deux adresses futures. Il faut néanmoins être conscient que cette machine ne peut tester que les configurations des variables d'entrée représentées à la figure 12a. Une quatrième instruction de la machine est l'instruction «do». Le répertoire d'instructions correspondant est représenté sur la figure 12b.

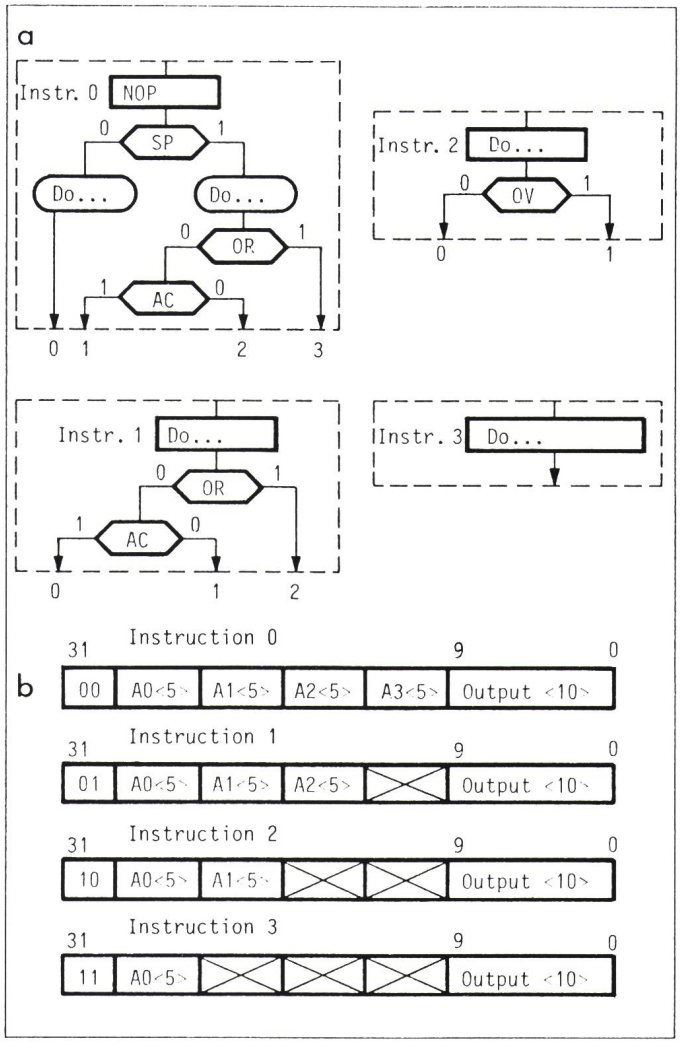
La figure 13a représente l'architecture de cette machine à 4-décision. Le programme obtenu ne comporte que 22 instructions, mais la ROM a été étendue à 32×32 bit, organisée en 16 lignes de 64 bit. Les 4 champs d'adresses sont connectés au multiplexeur d'adresses. La sortie de celui-ci est mémorisée dans un registre d'adresses de 5 bit. La figure 13b représente la microphotographie du circuit-test de cette machine à 4-décision avant son intégration avec le convertisseur A/D pour le circuit médical.

4. Comparaison

La machine à 4-décision de la section précédente nécessite, pour la com-

Figure 12
Quatre types d'instructions

a Organigrammes des instructions
b Formats des instructions 32 bits
‹ › contient le nombre de bit du champ en question.



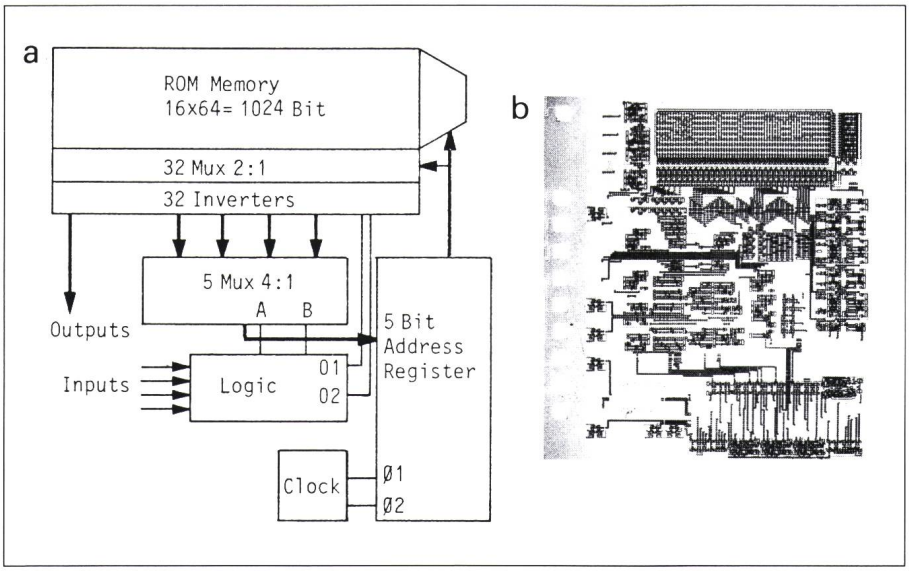


Figure 13 Machine à 4-décision industrielle

a Plan directeur
b Microphotographie

mande du convertisseur, 22 instructions de 32 bit, soit une ROM de 704 bit et une horloge de 10 kHz.

Si une machine de décision binaire avait été utilisée pour cette même fonction, celle-ci aurait nécessité 82 instructions de 18 bit, soit une ROM de 1476 bit et une fréquence de 40 kHz.

Si l'on compare la machine à 4-décision présentée à une MSS équivalente, on peut montrer que la ROM de cette dernière comporterait davantage de bit. Pour cet exemple particulier, le recours à une machine à N-décision est donc le meilleur choix.

De façon générale, le choix d'une machine à N-décision est recommandé si la machine comporte beaucoup d'entrées et que seules quelques entrées doivent être considérées pour déterminer l'état futur. Il est par contre avantageux de recourir à une MSS si la machine comporte un nombre limité d'entrées et que toutes ces entrées doivent être considérées pour déterminer l'état futur. C'est d'ailleurs le choix qui a été fait pour la MSS de la figure 2.

Les démarches menant aux machines Risc

Pour des microprocesseurs 32 bit, le recours à des architectures Risc a été proposé à la suite des analyses des programmes engendrés par des compilateurs et exécutés par ces machines. Il en résultait que la moitié environ des instructions machine n'étaient pas utilisées. Réduire le répertoire d'instructions de tels microprocesseurs en supprimant ces instructions inutiles n'était pas une mauvaise idée. Cela réduit la complexité de la conception du microprocesseur, et les performances en vitesse pourraient s'en trouver améliorées.

Cet article propose une autre démarche menant à des architectures de type Risc, appliquée, il est vrai, à des processeurs basse consommation et non pas à des machines 32 bit. Elle consiste à déterminer le répertoire d'instructions du processeur à partir des deux instructions fondamentales d'une machine de décision binaire, à savoir le *if...then...else* et le *do...* Très naturellement, on obtient alors un répertoire d'instructions réduit.

Encadré 1

Processeurs de type Risc

1. Machines de décision binaire et processeurs Risc

Un processeur peut être décomposé en une unité de traitement et une unité de commande. Pour un processeur rudimentaire, l'unité de commande peut être réalisée par une MSS ou une machine de décision binaire. Pour des microprocesseurs, l'unité de commande a pour rôle d'interpréter les instructions stockées dans une mémoire RAM externe. Pour des processeurs spécialisés, comme des montres électroniques ou des applications médicales, c'est l'application qui est directement programmée dans l'unité de commande. Si c'est une machine de décision binaire qui est utilisée, on programme l'application dans un langage de bas niveau, utilisant le répertoire d'instructions de la machine de décision binaire (fig. 8). On constate alors que le répertoire d'instructions est très limité, comme pour les processeurs Risc (Reduced Instruction Set Computers).

Cependant, le répertoire d'instructions de la figure 8 ne contient qu'une seule instruction de commande de l'unité de traitement, à savoir une instruction «do...». Pour un processeur spécialisé, comportant une unité de traitement avec une unité arithmétique et logique, il est nécessaire de pouvoir exécuter plusieurs opérations différentes. Le choix de celles-ci est réalisé par des instructions «do...» spécialisées comme l'addition, la soustraction, le chargement etc. C'est d'ailleurs ainsi que sont définis les répertoires d'instructions des microprocesseurs.

Il est dès lors évident que les répertoires d'instructions des processeurs spécialisés considérés comprendront de l'ordre de 20 instructions. Généralement, les instructions «do...» comprendront un champ pour définir l'opération que doit exécuter l'unité de traitement. De tels répertoires d'instructions ont donc les caractéristiques suivantes:

- un répertoire très limité
- chaque instruction est codée sur un seul mot
- toutes les instructions ont la même longueur
- chaque instruction est exécutée en un seul cycle, comportant un nombre fixe de phases
- le décodeur d'instructions est très simple et occupe une très faible partie de la surface de la puce, de l'ordre de 5%.

Toutes ces caractéristiques sont bien connues comme étant celles, entre autres, des processeurs Risc [17; 18; 19].

On peut d'ailleurs observer que l'application des machines de décision binaire pour des processeurs basse consommation a été étudiée [20; 21] sans avoir eu au préalable connaissance des machines Risc pour des microprocesseurs. Les deux approches ont mené à des répertoires d'instructions limités, mais pour des raisons différentes. Si la vitesse est le premier critère pour les machines Risc, c'est la basse consommation qui a dicté ce choix pour les processeurs horlogers [22]. En effet, un répertoire d'instructions limité, ainsi qu'une exécution d'instruction en un seul cycle conduisent à un nombre minimal de coups d'horloge pour une tâche donnée et ainsi à une très faible consommation, le processeur étant arrêté dès que la tâche est terminée. C'est pour cette raison que nous avons appelé ces machines des processeurs Risc basse consommation [22].

2. Processeurs Risc basse consommation

Des architectures Risc basse consommation ont été utilisées pour des processeurs horlogers [6; 22], décrites dans la section prochaine. De telles architectures comportent une unité arithmétique et logique rudimentaire et une mémoire RAM de quelques mots. L'unité de commande est une machine de décision binaire dont la schéma est représenté à la figure 7.

Il est bien évident que les variables à tester par la machine de décision binaire ne proviennent pas seulement des circuits périphériques mais aussi de l'unité de traitement. De plus, certains booléens doivent pouvoir être mis à «1» ou à «0» et testés par l'unité de commande.

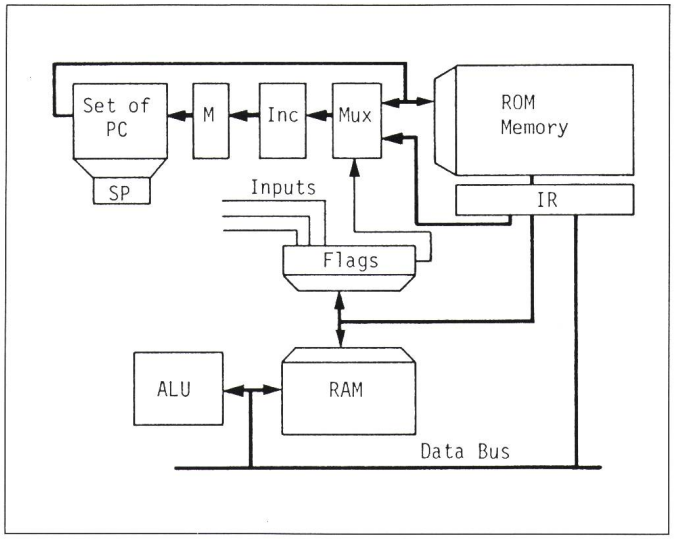
La figure 14 représente l'architecture d'un processeur Risc basse consommation. Une différence existe par rapport à la figure 7: le multiplexeur de test a été remplacé par une banque de booléens adressés par le même décodeur que celui de la mémoire RAM. Cela implique bien évidemment que le nombre de booléens est le même que

celui des registres RAM. De plus, certains signaux provenant de circuits périphériques sont liés directement à quelques-uns de ces booléens.

Ces processeurs comportent généralement plus de 1000 instructions. Il est donc intéressant de constater qu'industriellement, les machines de décision binaire ne sont utilisées que pour des tâches importantes nécessitant un grand nombre d'instructions.

Des travaux de recherches [20; 21] ont montré qu'un processeur horloger pouvait être réalisé sans unité arithmétique et logique. L'idée est de remplacer les opérations arithmétiques, essentiellement l'incrémentation et la décrémentation, par des algorithmes de décision binaire [13]. Cela n'a pas été exploité pour des réalisations industrielles, d'une part parce que la conception d'une unité arithmétique est très aisée et, d'autre part, parce que le nombre d'instructions exécutées est moindre, ce qui est favorable sur le plan de la consommation. Cependant, de nombreuses parties du programme d'un processeur horloger furent réalisées par des algorithmes de décision binaire, comme par exemple la détermination du nombre de jours d'un mois [21; 23]. Cela illustre un choix fondamental, à savoir d'implémenter telle ou telle fonction en logiciel ou en matériel [13; 24].

Figure 14
Architecture d'une machine Risc basse consommation

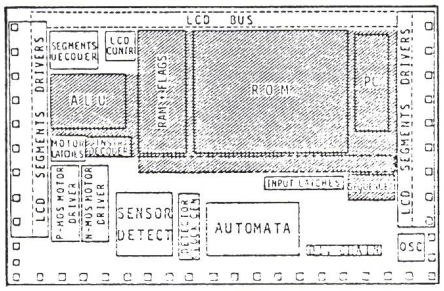


3. Processeurs horlogers industriels

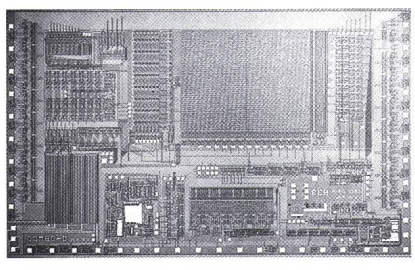
La figure 15a représente le plan directeur d'un processeur horloger [6] dont l'architecture, sans les circuits périphériques, est représentée à la figure 14. Ce processeur ne comporte que 12 instructions de 16 bit (fig. 15c). La mémoire ROM contient 800 instructions, soit 12 800 bit. La mémoire RAM comporte 15 registres de 7 bit et 15 booléens. Le nombre total de transistors est d'environ 20 000.

Ce processeur ne comporte pas de pile matérielle pour les sous-programmes. Cependant, un niveau de sous-programme est utilisé avec un retour de sous-programme réalisé en logiciel [13]. Le mécanisme est basé sur le test d'un registre de donnée caractéristique du point de retour, pour calculer l'adresse de retour au lieu de la mémoriser. Cela illustre une nouvelle fois l'équivalence entre le logiciel et le matériel. Le décodeur d'instructions et le séquenceur, comme cela est le cas pour des machines Risc, n'occupent qu'environ 5% de la surface de la puce. La MSS décrite au début de cet article est utilisée comme périphérique dans ce circuit. La figure 15b représente la mi-

a



b



c

Instruction		Code															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	JPZ, JPNZ	0	Branch Address										P	Flag			
2	JMP	0	Branch Address										0	1	1	1	1
3	Data→RAM	1	0	0	1	H	Data (7 bits)							RAM			
4	AC→RAM	1	0	0	0	H	0	0	0	0	0	0	0	RAM			
5	AC→(OP)RAM	1	1	1	0	H	0	0	0	0	0	OP		RAM			
6	AC=Data	1	1	0	1	H	Data (7 bits)							0	0	0	0
7	SR Flags	1	0	1	1	H	0	0	0	0	0	0	SR	Flag			
8	Motor	1	1	0	0	H	0	0	0	R	M1	MC	M2	0	0	0	0
9	Reset	1	0	1	0	H	0	0	0	0	HZ	ED	0	0	0	0	0
10	NOP	1	0	1	0	H	0	0	0	0	0	0	0	0	0	0	0
11	Num Output	1	1	1	1	H	0	SUP	COL	Flash G	D	Digit		RAM			
12	Alpha Out	1	1	1	1	H	1	SUP	COL	Flash G	D	Digit		RAM			

Figure 15 Processeur horloger

a Plan directeur
b Microphotographie
c Répertoire d'instructions
R Moteur run
M1, MC, M2 Commande moteur
HZ Base de temps
ED Entrée données

crophotographie de ce circuit. D'autres processeurs horlogers ont été intégrés, le plus complexe atteignant environ 35 000 transistors [15; 22].

Conclusion

Il est certain qu'aujourd'hui, la plupart des unités de commande relativement simples sont conçues en logique câblée. Les outils CAO, comme la technique des Cellules standards, ainsi que la formation en logique de base, en sont responsables. Cependant, il est évident que la plupart du temps une unité de commande programmée est meilleure. Leur conception est plus structurée de par la programmation de la mémoire ROM, avec les avantages inhérents comme la facilité de correction et de modification. Cet article a montré que toute une gamme d'unités de commande programmées peuvent être utilisées industriellement.

Dans une courbe en fonction de la complexité des machines, les machines de décision binaire se situent à mi-chemin entre les MSS et les microprocesseurs. Elles ne sont utilisées que pour des tâches complexes nécessitant plus de 500 instructions et, dans ce cas, elles sont nettement moins coûteuses que d'autres réalisations.

La théorie sur les machines de décision binaire a pour but de réaliser en logiciel des fonctions réalisées traditionnellement en matériel et de montrer l'équivalence entre le logiciel et le matériel. En pratique, les machines de décision binaire sont utilisées industriellement comme unités de commande de processeurs horlogers. L'architecture du processeur qui en résulte comporte alors plusieurs caractéristiques des machines Risc, avec les avantages en consommation cités plus haut.

On observe d'autre part sur le marché que plusieurs circuits intégrés horlogers ont adopté une architecture très proche des microprocesseurs 8 bit classiques, plutôt de type Cisc (Complex Instruction Set Computer). Un débat Risc-Cisc peut ainsi également avoir lieu pour des circuits intégrés à applications spécifiques.

Literatur

[1] *A.J. Kessler* and *A. Ganesan:* Standard cell VLSI design: A tutorial. IEEE Circuits and Devices Magazine 1(1985)1, p. 17...34.

[2] *F. Anceau* and *R.A. Reis:* Complex integrated circuit design strategy. IEEE J. Solid-State Circuits 17(1982)3, p. 459...464.

[3] *M. Obrebska:* Efficiency and performance comparison of different design methodologies for control parts of microprocessors. Microprocessing and Microprogramming 10(1982)-, p. 163...178.

[4] *Z. Kohavi:* Switching and finite automata theory. Second edition. New York, McGraw-Hill, 1978.

[5] *C. Clare:* Designing logic systems using state machines. New York, McGraw-Hill, 1973.

[6] *J.F. Perotto:* Circuit horloger à structure de processeur. Bull. ASE/UCS 75(1984)11, p. 620...624.

[7] *J.-C. Martin:* Random and programmable logic in watches. In: From electronics to microelectronics. Fourth European Conference on Electrotechnics - Eurocon '80 p. 708...714.

[8] *C. Piguet* a.o.: A metal-oriented layout structure for CMOS logic. IEEE J. Solid-State Circuits 19(1984)3, p. 425...436.

[9] *D.A. Mange:* A high-level-language programmable controller. Part. I: A controller for structured microprogramming. Part II: Microcompilation of the high-level-language micropascal. IEEE Micro 6(1986)1, p. 25...41 + Nr. 2, p. 47...63.

[10] *P.J.A. Zsombor-Murray* a.o.: Binary-decision-based programmable controllers. Part I/II/III. IEEE Micro 3(1983)4, p. 67...83, Nr. 5, p. 16...26 + Nr. 6, 24...39.

[11] *R.T. Boute:* The binary-decision machine as programmable controller. Euromicro Newsletter 2(1976)1, p. 16...22.

[12] *E. Cerny, D. Mange* and *E. Sanchez:* Synthesis of minimal binary decision trees. IEEE Trans. C 28(1979)7, p. 472...482.

[13] *D. Mange:* Compteurs microprogrammés. Bull. ASE/UCS 70(1979)19, p. 1087...1095.

[14] *C. Piguet:* Design methodology for full custom CMOS microcomputers. Integration, the VLSI journal, 1(1983)-, p. 335...350.

[15] Scientific report 1986. Neuchâtel, Centre Suisse d'Electronique et de Microtechnique (CSEM), 1986.

[16] *L.D. Coraor, P.T. Hulina* and *O.A. Morean:* A general model for memory-based finite-state machines. IEEE Trans. C 36(1987)2, p. 175...184.

[17] *A. Ditzinger* and *J. Beister:* Word reduction in microprogrammed controllers by state-dependent preprocessing of the inputs. Microprocessing and Microprogramming 9(1982)-, p. 161...173.

[18] *D.T. Fitzpatrick* a.o.: A RISCy approach to VLSI. VLSI Design -(1981)4, p. 14...20 or Computer Architecture News 10(1982)1, p. 28...32.

[19] *D.A. Patterson* and *C.H. Séquin:* A VLSI risc. IEEE Computer 15(1982)9, p. 8...21.

[20] *C. Piguet* a.o.: Microcomputer design for digital watches. In: Implementing functions: microprocessors and firmware. Seventh Euromicro Symposium on Microprocessing and Microprogramming, Paris, 1981; p. 431...442.

[21] *E. Sanchez* et *A. Stauffer:* Horloge microprogrammée. 10e Congrès International de Chronometrie (CIC), Genève, 11 au 14 septembre 1979. Actes; p. 279...284.

[22] *M. Ansorge, C. Piguet* and *E. Dijkstra:* Design methodology for low power full custom RISC microprocessors. Proceedings of the Euromicro 1986; p. 427...434.

[23] *D. Mange, E. Sanchez* and *A. Thayse:* Binary-decision-based programmable controllers. IEEE Micro 5(1985)3, p. 58...72.

[24] *R. Bernhard:* More hardware means less software. IEEE Spectrum 18(1981)12, p. 30...37.