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: 70 (1979)

Heft: 19

Artikel: Compteurs microprogrammés

Autor: Mange, D.

DOI: https://doi.org/10.5169/seals-905434

Compteurs microprogrammés

Par D. Mange

621.317.785::681.326.32

Tout compteur peut être décrit par un organigramme, c'est-à-dire par un assemblage d'instructions de test et de sortie. La mise en évidence d'un sous-programme produit des modes de représentation équivalents à l'organigramme original et réalisables par un arbre de décision binaire (solution logicielle) ou par une pile (solution matérielle). Ces diverses variantes sont appliquées à deux exemples concrets, tirés de l'étude d'une horloge digitale: un compteur par six et un compteur par soixante. Les microprogrammes et les réalisations matérielles découlant de ces variantes sont analysés et comparés tant au point de vue du coût (capacité de la mémoire) que de la rapidité.

Jeder Zähler kann durch ein Organigramm dargestellt werden, d.h. durch die Verbindung von Prüf- und Ausgangsinstruktionen. Wird ein Unterprogramm herangezogen, so entstehen gleichwertige Darstellungsarten zum ursprünglichen Organigramm, die mit einem binären Entscheidungsbaum (Software-Lösung) oder einem Stapel (Hardware-Lösung) ausgeführt werden können. Diese verschiedenen Möglichkeiten werden auf zwei Beispiele aus einer Studie über eine Digitaluhr angewandt: auf einen 6er und einen 60er Zähler. Die entsprechenden Mikroprogramme und materiellen Verwirklichungen werden untersucht und bezüglich Kosten (Speicherkapazität) und Geschwindigkeit verglichen.

1. Introduction

1.1 Préambule

On a montré dans une publication précédente [1] que tout système logique combinatoire peut être représenté par un arbre ou par un algorithme de décision binaire. Un tel arbre (ou un tel algorithme) est toujours réalisable par un microprogramme enregistré dans la mémoire d'un système séquentiel: la machine de décision binaire. Dans le présent article, on étendra les applications de cette machine à la réalisation des compteurs ou diviseurs de fréquence. Deux exemples, découlant de l'étude d'une horloge digitale, mettront en évidence diverses méthodes de microprogrammation et introduiront notamment la notion de sous-programme.

Un accent particulier est mis sur les interactions entre les réalisations matérielles (système logique câblé) et logicielles (système logique programmé), suggérant ainsi une éventuelle unification dans la méthodologie de l'informatique.

1.2 Organigramme

Un *organigramme* (fig. 2 et 4a, par ex.) est un mode de représentation des systèmes logiques, constitué par l'assemblage de deux types d'éléments ou *instructions:*

– une instruction de *test* (en anglais «if ... then ... else»), représentée par un losange; chaque instruction de test est définie par un *nombre octal* ou *adresse i,* une variable de test *a,* une entrée (provenant d'une ou plusieurs instructions précédentes) et deux sorties ($a = 1$, $a = 0$) conduisant chacune à une instruction suivante $i^+(a = 1)$ et $i^+(a = 0)$; la sortie de la variable complémentée ($a = 0$) est repérée par un *rond* accolé au losange;

– une instruction de *sortie,* représentée par un rectangle; chaque instruction de sortie est définie par son adresse i, un état de sortie Z, une entrée (provenant d'une ou plusieurs instructions précédentes) et une sortie conduisant à une instruction suivante i^+.

Les règles d'assemblage précisent que:

– une sortie d'une instruction quelconque n'est reliée qu'à une seule entrée d'une instruction suivante;

– une entrée d'une instruction quelconque peut être reliée aux sorties de plusieurs instructions précédentes.

Les *algorithmes* et les *arbres de décision binaire* [1] sont donc des organigrammes particuliers.

Pour la représentation graphique des organigrammes, on a utilisé en principe les conventions proposées dans la référence [2].

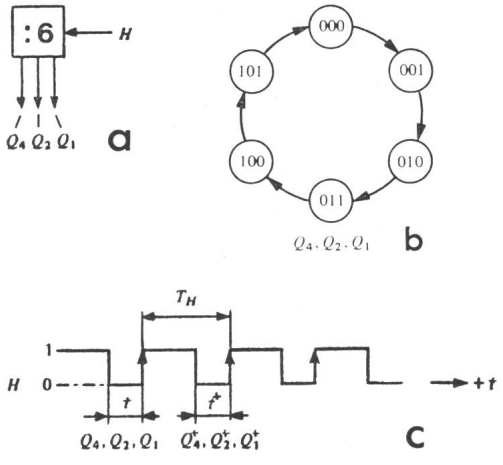


Fig. 1 Compteur par six
a Schéma
b Graphe des états
c Variations de l'horloge H

2. Graphe des états et organigramme linéaire

2.1 Exemple: compteur par six

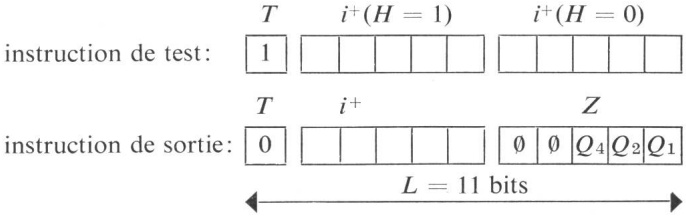
Un compteur par six, commandé par une horloge extérieure H (fig. 1a), doit produire la séquence suivante, dans le code binaire pur:

$$Q_4, Q_2, Q_1 = 000 \rightarrow 001 \rightarrow 010 \rightarrow 011 \rightarrow 100 \rightarrow 101 \rightarrow 000 \qquad (1)$$

Dans le graphe des états de la figure 1b, on sous-entend le rôle de la variable H (fig. 1c), dont chaque montée ($H = 0 \rightarrow 1$) provoque le passage d'un état présent Q_4, Q_2, Q_1 à l'état futur Q_4^+, Q_2^+, Q_1^+ [3, pp. 129...131].

2.2 Organigramme linéaire: programmation

L'organigramme de la fig. 2 réalise le compteur par six précédent; on appellera *organigramme linéaire* ce mode de représentation, qui découle directement du graphe des états. Les 18 instructions (00 à 21 en octal) nécessitent 5 bits d'adresse, car $18 < 32 = 2^5$; on en déduit le format (en binaire) des deux instructions de test et de sortie, où ∅ représente une *condition indifférente* (0 ou 1):



La liste des instructions constitue le *programme* ou *microprogramme* du compteur; la table I représente le programme, en octal, réalisant l'organigramme de la fig. 2.

2.3 Capacité de la mémoire

Appelons L la largeur d'une instruction [bit], n le nombre total des instructions d'un programme et $C = nL$ [bit] le nombre total des bits à mémoriser ou *capacité de la mémoire*.

Dans le cas général d'un comptage par p, l'organigramme linéaire nécessite $n = 3p$ instructions (fig. 2) et, par conséquent, $\lceil \log_2 3p \rceil$ bits d'adresse, où la notation $\lceil x \rceil$ désigne le plus petit entier supérieur ou égal à x [6, p. 37]. La largeur L d'une instruction est donc égale à $1+2\lceil \log_2 3p \rceil$ [bit] et la capacité de la mémoire vaut alors

$$C(p) = nL = 3p\left(1+2\lceil \log_2 3p \rceil\right) \quad \text{[bit]} \qquad (2)$$

On retrouve dans l'exemple du compteur par six $C(6) = 18 \cdot 11 = 198$ bits!

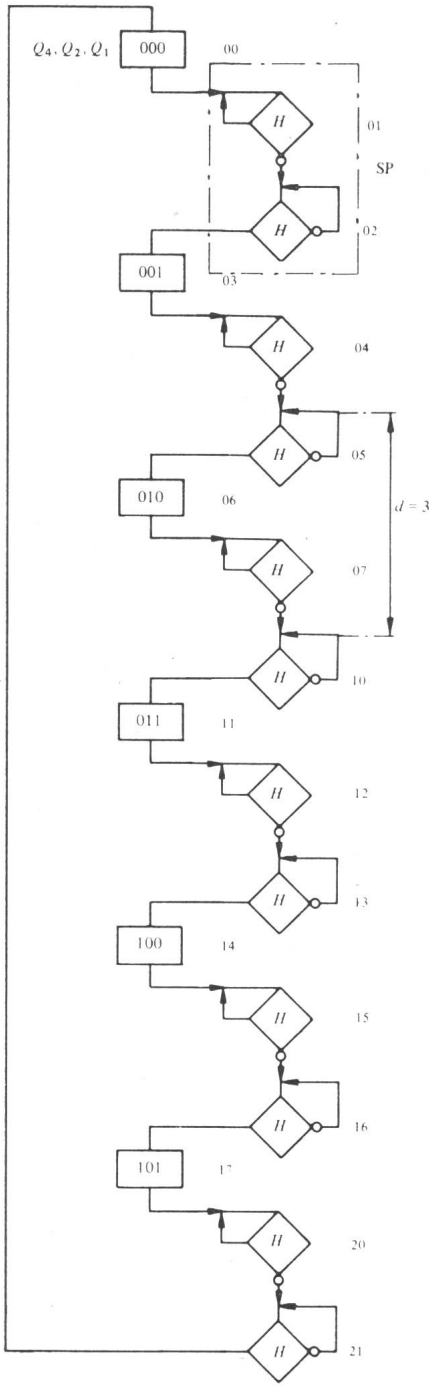


Fig. 2 Organigramme linéaire du compteur par six
SP Sous-programme
d Durée d'un cycle

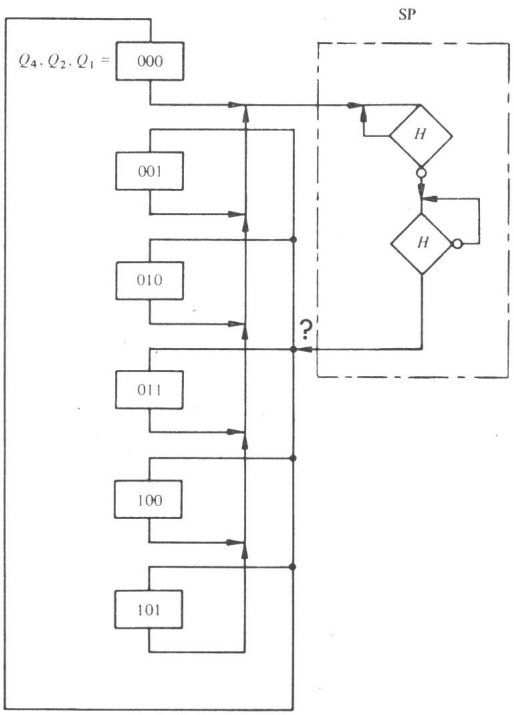


Fig. 3 Mise en évidence du sous-programme SP dans l'organigramme linéaire de la figure 2

2.4 Durée d'un cycle

Le nombre des instructions séparant deux montées successives de la variable H est appelé *durée d'un cycle* d (fig. 2: $d = 3$). Dans ce qui suit, on suppose que la période T_{CK} du signal d'horloge interne CK de la machine de décision binaire (fig. 6) est toujours inférieure à la période T_H de la variable extérieure H (fig. 1c), c'est-à-dire que

$$d \cdot T_{CK} < T_H \quad \text{[s]} \qquad (3)$$

La durée d'un cycle d est donc une mesure de la rapidité du système logique programmé: plus d est faible, plus faible sera la fréquence de l'horloge interne CK pour une même fréquence de la variable H.

Dans le cas de l'organigramme linéaire (fig. 2), la durée d'un cycle d est indépendante de p et vaut $d(p) = d(6) =$ constante $= 3$.

2.5 Réalisation matérielle

Une machine de décision binaire, semblable à celle décrite dans [1, fig. 6], peut réaliser le programme de la table I, à condition d'effectuer les adaptations matérielles suivantes:

- assurer les connexions $A_1 = B_1 = A_0 = B_0 = H$;
- étendre la capacité de la mémoire et du registre d'instructions de 4 à 5 bits d'adresse (i_4, i_3, i_2, i_1, i_0);
- lire l'état de sortie en faisant Z_2, Z_1, $Z_0 = Q_4$, Q_2, Q_1.

La machine de décision binaire décrite à la fig. 6 du présent article peut également réaliser le même programme si l'on effectue les adaptations suivantes:

- pour le matériel, assurer les connexions $A_0 = A_1 = A_2 = A_3 = H$;
- pour le logiciel, programmer les valeurs suivantes (en binaire): a_1, $a_0 = \emptyset\emptyset$ pour $T = 1$; a_1, $a_0 = 01$ pour $T = 0$ (les sorties Q_4, Q_2, Q_1 du compteur sont celles du registre REG 1).

3. Sous-programme: réalisation par un arbre de décision

3.1 Sous-programme

Tout organigramme comportant une entrée unique et une sortie unique est un *sous-programme.*

L'organigramme composé des deux instructions 01 et 02 (fig. 2) est donc un sous-programme (SP). Ce sous-programme se répète six fois dans la fig. 2; dans le but de simplifier le programme final, on est donc tenté de transformer l'organigramme original par la mise en évidence du sous-programme SP (fig. 3). Mais l'assemblage d'instructions ainsi obtenu n'est plus un organigramme: la sortie du sous-programme est reliée aux entrées de plusieurs instructions suivantes, en violation de la première règle d'assemblage. Deux solutions seront exposées ici pour lever cette contradiction: l'une faisant appel à un arbre de décision binaire (solution logicielle), l'autre introduisant une pile (solution matérielle).

Programme octal réalisant l'organigramme linéaire de la fig. 2 Table I

i	T	$i^+(H=1)$ ou i^+	$i^+(H=0)$ ou $Z=Q_4,Q_2,Q_1$
00	0	01	00
01	1	01	02
02	1	03	02
03	0	04	01
04	1	04	05
05	1	06	05
06	0	07	02
07	1	07	10
10	1	11	10
11	0	12	03
12	1	12	13
13	1	14	13
14	0	15	04
15	1	15	16
16	1	17	16
17	0	20	05
20	1	20	21
21	1	00	21

Programme octal réalisant l'organigramme de la fig. 4a Table II

i	T	a	$i^+(a=1)$ ou i^+	$i^+(a=0)$ ou $Z=Q_4,Q_2,Q_1$
00	0	–	01	00
01	1	0	01	02
02	1	0	03	02
03	1	3	07	04
04	1	2	06	05
05	1	1	11	10
06	1	1	13	12
07	1	1	00	14
10	0	–	01	01
11	0	–	01	02
12	0	–	01	03
13	0	–	01	04
14	0	–	01	05

Codage binaire des variables de test Table III

a	$a_1\ a_0$
H	0 0
Q_1	0 1
Q_2	1 0
Q_4	1 1

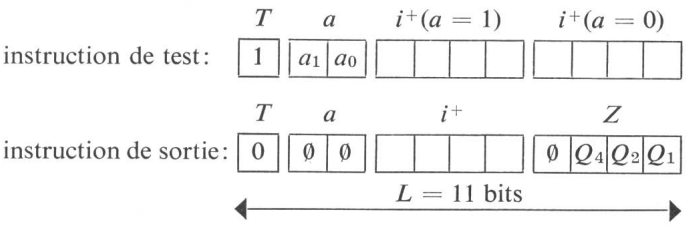
3.2 Arbre de décision binaire

L'état présent du compteur ($Q_4, Q_2, Q_1 = 000$ par ex.) détermine complètement son état futur ($Q_4^+, Q_2^+, Q_1^+ = 001$). Un arbre de décision binaire [1], dont les variables de test sont Q_4, Q_2 et Q_1, permet alors de gagner, à la sortie du sous-programme SP (fig. 4a), l'état futur adéquat. La table de *Karnaugh* de la fig. 4b justifie le calcul de l'arbre minimal de la fig. 4a: une simplification a été rendue possible par la présence de deux états $\emptyset$ ($Q_4, Q_2, Q_1 = 110$ et 111) [4].

L'assemblage d'instructions de la figure 4a est un organigramme. Au prix d'un certain logiciel (les instructions de test de l'arbre), on a conservé l'unique sous-programme SP, tel qu'il a été mis en évidence dans la fig. 3.

3.3 Programmation

Les treize instructions de la fig. 4a nécessitent 4 bits d'adresse, tandis que le test des quatre variables d'entrée $(a \in \{H, Q_1, Q_2, Q_4\})$ requiert 2 bits de codage a_1, a_0 (table III). On en déduit le format (en binaire) des deux instructions de test et de sortie:



La table II représente le programme, en octal, de l'organigramme de la figure 4a.

Dans le cas général d'un comptage par p, on dénombre p instructions de sortie, $p-1$ instructions de test dans l'arbre minimal et deux instructions de test du sous-programme, soit $n = 2p+1$ instructions au total, nécessitant $\lceil \log_2(2p+1) \rceil$ bits d'adresse. Le nombre des variables d'entrée a est égal à

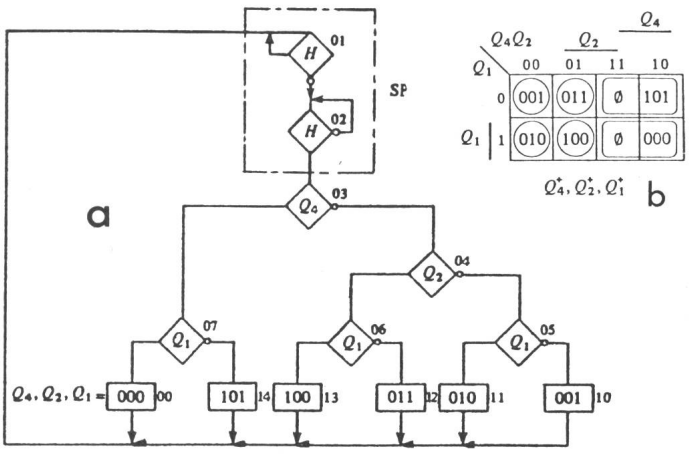


Fig. 4 Sous-programme SP avec un arbre de décision binaire (a) et simplification de cet arbre de décision (b)

$1 + \lceil \log_2 p \rceil$ et, par conséquent, le nombre des bits de codage a_i est égal à $\lceil \log_2 (1 + \lceil \log_2 p \rceil) \rceil$. La capacité de la mémoire $C(p)$ vaut dans ce cas:

$$C(p) = nL = (2p + 1) \left[1 + \lceil \log_2 (1 + \lceil \log_2 p \rceil) \rceil + 2 \lceil \log_2 (2p + 1) \rceil \right] \text{ [bit]} \quad (4)$$

qui devient dans l'exemple traité $C(6) = 13 \cdot [1 + 2 + 2 \cdot 4] = 143$ bits.

La durée d'un cycle $d(p)$ dépend des dimensions de l'arbre (fig. 4a) et vaut, dans le pire des cas (arbre complet):

$$d(p) = 3 + \lceil \log_2 p \rceil \quad (5)$$

Dans l'exemple traité, on a donc: $d(6) = 6$.

3.4 Réalisation matérielle

A condition de poser $A_1 = H$, $B_1 = Q_1$, $A_0 = Q_2$, $B_0 = Q_4$ et $Z_2, Z_1, Z_0 = Q_4, Q_2, Q_1$, la machine de décision binaire décrite dans [1, fig. 6] peut réaliser le programme de la table II.

La machine de décision binaire de la fig. 6 peut également réaliser le même programme, à condition d'effectuer les adaptations suivantes:

- pour le matériel, assurer les connexions $A_0 = H$, $A_1 = Q_1$, $A_2 = Q_2$, $A_3 = Q_4$;
- pour le logiciel, programmer les valeurs suivantes (en binaire) pour les instructions de sortie ($T = 0$): $a_1, a_0 = 01$ (les sorties Q_4, Q_2, Q_1 du compteur sont celles du registre REG 1).

3.5 Commentaire

La synthèse d'un compteur synchrone câblé, donné par le graphe de la fig. 1b, peut être entreprise à partir de la table de *Karnaugh* de la fig. 4b [3, pp. 135–137]. En supposant l'emploi de bascules D, on vérifie les relations

$$Q_4^+ = D_4, \; Q_2^+ = D_2, \; Q_1^+ = D_1 \quad (6)$$

où D_4, D_2 et D_1 sont les variables d'excitation des trois bascules. L'arbre de décision minimal de la fig. 4a (instructions de test 03 à 07) est donc équivalent à l'arbre de démultiplexeurs minimal réalisant les fonctions D_4, D_2 et D_1 du compteur câblé [4]; un tel compteur est caractérisé par un codage des états minimal [3, p. 205].

Par analogie, l'organigramme linéaire de la fig. 2 est équivalent au logigramme réalisant le compteur synchrone câblé dans un codage 1 parmi M (compteur en anneau) [3, p. 142, p. 211]; chaque groupe de trois instructions successives (une instruction de sortie et les deux instructions de test du sous-programme) est équivalent à l'une des six bascules bistables D du compteur câblé, commandées par l'horloge H.

Relevons enfin que, dans le cas de l'organigramme linéaire (fig. 2), toute l'information du graphe des états est contenue dans le programme; dans le cas de l'arbre (fig. 4a), la machine de décision binaire doit tester les valeurs des sorties Q_4, Q_2 ou Q_1: il existe dans ce cas une rétroaction matérielle entre le registre de sortie (fig. 6: REG 1) et les entrées des variables de test (fig. 6: A_1, A_2, A_3).

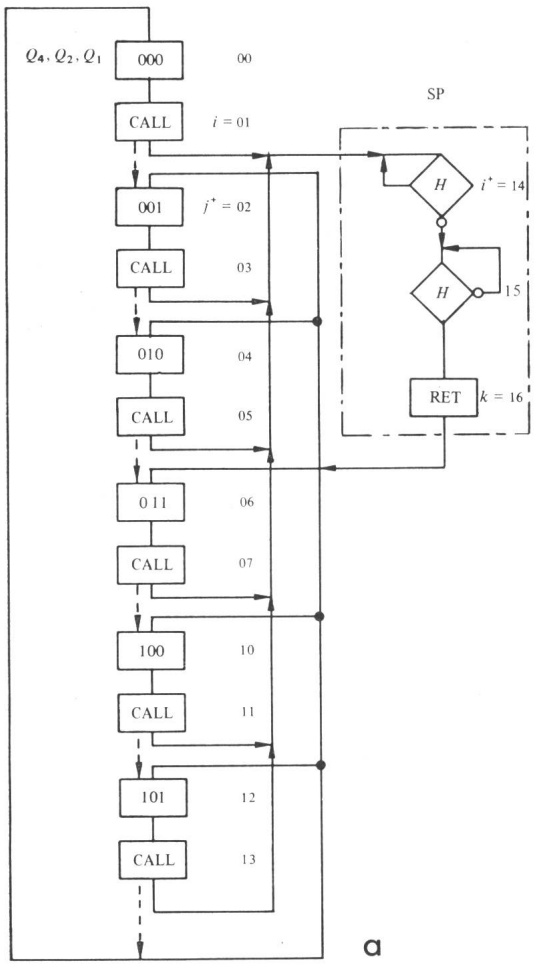


Fig. 5a Réalisation de principe d'un sous-programme avec une pile

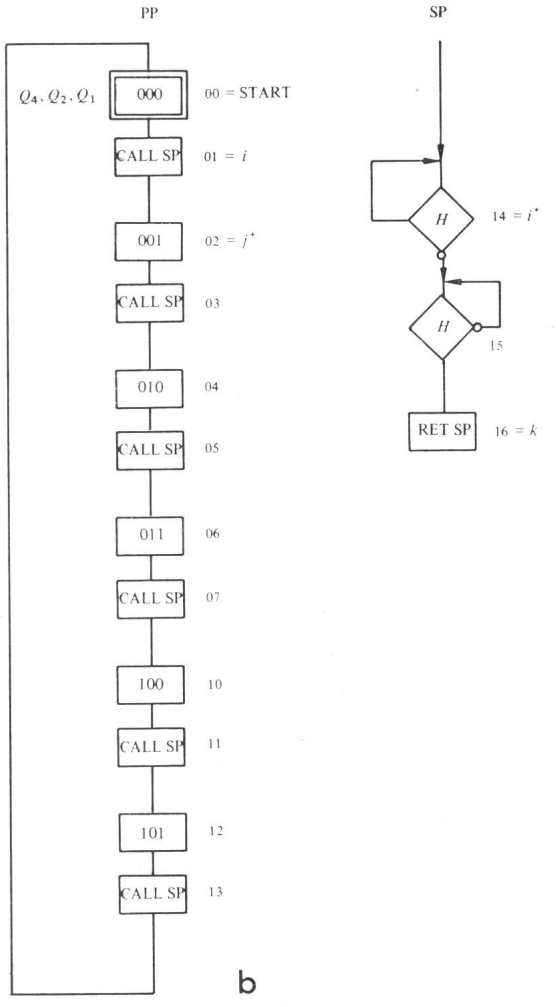


Fig. 5b Organigramme du programme principal (PP) et du sous-programme (SP) réalisé par une pile

4. Sous-programme: réalisation par une pile (stack)

4.1 Pile

On a déjà vu que l'état présent du compteur (Q_4, Q_2, $Q_1 = 000$ par ex.) détermine complètement l'état futur (Q_4^+, Q_2^+, $Q_1^+ = 001$). Un registre de sortie particulier, la *pile* (en anglais stack ou LIFO: last in, first out register), mémorise l'adresse j^+ de l'état futur (fig. 5a) au moment de l'appel du sous-programme, et restitue cette adresse après l'exécution de celui-ci.

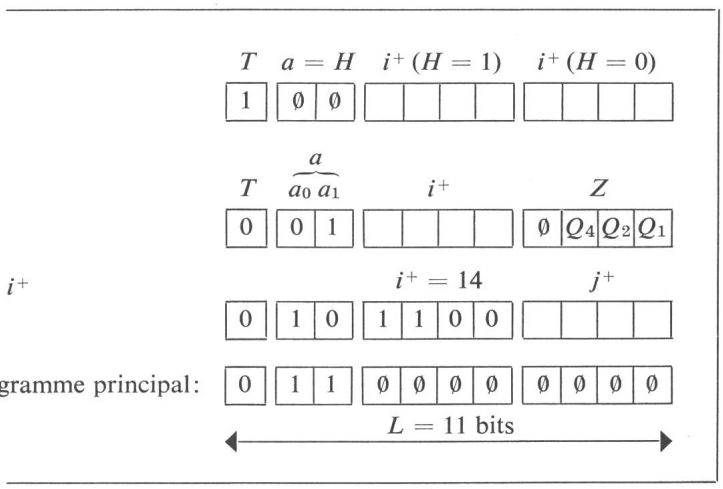
Le logiciel d'une telle opération nécessite l'introduction de deux instructions de sortie particulières:

- une instruction CALL ou *instruction d'appel* ($i = 01$, par ex.), appelant le sous-programme ($i^+ = 14$) et mémorisant dans la pile l'adresse de l'état futur ($j^+ = 02$);
- une instruction RET ou *instruction de retour* ($k = 16$), terminant le sous-programme, restituant l'adresse de l'état futur ($j^+ = 02$) et exécutant le retour du sous-programme dans le programme principal, à cette dernière adresse.

L'assemblage d'instructions de la fig. 5a ne respecte pas la première règle régissant les organigrammes. La fig. 5b présente alors deux organigrammes distincts, équivalents à l'assemblage de la fig. 5a et qui mettent en évidence le programme principal (PP) et le sous-programme (SP).

4.2 Programmation

Les quinze instructions de la fig. 5b nécessitent 4 bits d'adresse; le test de l'unique variable d'entrée H n'implique aucun codage, tandis que la distinction des trois types d'instructions de sortie (affichage de l'état du compteur Q_4, Q_2, Q_1, CALL et RET) requiert 2 bits a_1, a_0; il en découle le format (en binaire) des quatre instructions suivantes:



La table IV présente le programme octal tiré de la fig. 5b.

Un compteur par p nécessite $2p$ instructions dans le programme principal et trois instructions dans le sous-programme, soit $n = 2p + 3$ instructions au total, impliquant l'emploi de $\lceil \overline{\log_2(2p + 3)} \rceil$ bits d'adresse. Les trois types d'instructions de sortie entraînent l'existence de 2 bits de codage a_1, a_0. La capacité de la mémoire $C(p)$ est égale dans ce cas à:

$$C(p) = nL = (2p + 3)\left[3 + 2\lceil \overline{\log_2(2p + 3)} \rceil\right] \quad \text{[bit]} \qquad (7)$$

qui, dans l'exemple traité, prend la valeur $C(6) = 15 \cdot [3 + 2 \cdot 4] = 165$ bits.

La durée d'un cycle d est indépendante de p et vaut: $d(p) = d(6) = \text{constante} = 5$.

4.3 Réalisation matérielle

La machine de décision binaire de la fig. 6 est conçue pour exécuter chacune des quatre instructions du paragraphe 4.2 selon les modalités suivantes:

- pour l'instruction de test ($T = 1$), on assure les connexions $A_0 = A_1 = A_2 = A_3 = H$;
- pour l'instruction de sortie affichant l'état Q_4, Q_2, Q_1 (T, a_1, $a_0 = 001$), les variables de sortie sont celles du registre REG 1;
- pour l'instruction CALL (T, a_1, $a_0 = 010$), le démultiplexeur DEMUL produit une variable logique $PUSH = 1$ qui mémorise l'adresse j^+ dans le registre SP1 de la pile;
- pour l'instruction RET (T, a_1, $a_0 = 011$), le démultiplexeur DEMUL produit une variable $POP = 1$ qui transfère le contenu du registre SP 1 de la pile (adresse j^+) dans le registre d'instructions REGINS, par l'intermédiaire du multiplexeur MUL 2.

Le signal $\overline{CLR}$ assure la remise à zéro de tous les registres (REGINS, REG 1, REG 0), ainsi que ceux de la pile (SP 1, SP 2). L'instruction initiale du programme (fig. 5b: $i = 00 =$ START) doit appartenir au programme principal, et non au sous-programme; dans ce dernier cas, une remise à zéro générale effacerait le contenu des registres de la pile et empêcherait le retour correct du sous-programme dans le programme principal.

4.4 Commentaire

Le registre SP 2 de la pile (fig. 6) permet la réalisation d'un deuxième niveau de sous-programme, comme le démontre l'application du paragraphe 6.4. En toute généralité, une pile comportant m registres identiques SP 1...SP_m, connectés de façon itérative, réalise m niveaux de sous-programmes.

En conclusion, on dira que la réalisation matérielle d'une pile est équivalente à la programmation d'un arbre de décision binaire (fig. 4a); la clarté du programme obtenu (fig. 5b), la valeur constante de la durée du cycle d et l'absence de rétroactions matérielles (du registre de sortie aux entrées des variables de test) sont les principaux avantages de cette solution.

5. Diviseur de fréquence et arbre de décision décomposé

5.1 Réalisation câblée

Selon la terminologie adoptée en [3, p. 118], un *diviseur de fréquence* est un compteur dont toutes les bascules sont des diviseurs de fréquence par deux, caractérisés par l'équation

Programme octal réalisant les organigrammes de la fig. 5b Table IV

i	T	a	$i^+ (H = 1)$ ou i^+	$i^+ (H = 0)$ ou Z ou j^+
00	0	1	01	00
01	0	2	14	02
02	0	1	03	01
03	0	2	14	04
04	0	1	05	02
05	0	2	14	06
06	0	1	07	03
07	0	2	14	10
10	0	1	11	04
11	0	2	14	12
12	0	1	13	05
13	0	2	14	00
14	1	–	14	15
15	1	–	16	15
16	0	3	–	–

$Q^+ = \overline{Q}$ [3, pp. 106–107]. La mise en série de trois bascules de ce type (fig. 7a, avec $\overline{CLR}_2 = 1$) produit un diviseur par huit, tandis que la réalisation d'une fonction de rétroaction $Z_6 = Q_4\,Q_2 = CLR_2$ (fig. 7b) produit un diviseur par six dont le graphe des états permanents est celui de la fig. 1b.

5.2 Programmation

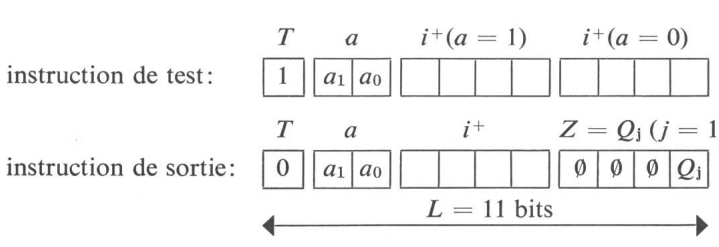
L'organigramme de la fig. 7c représente un diviseur de fréquence par six équivalent à celui de la fig. 7a:

– chacun des diviseurs de fréquence par deux (la bascule bistable Q_1, par ex.) est réalisé par une instruction de test ($i = 03$) et deux instructions de sortie ($i = 04$ et $i = 10$) exécutant l'équation $Q^+ = \overline{Q}$; de plus, toute descente d'une bascule ($Q_1 = 1 \rightarrow 0$ par ex.) entraîne une variation de l'état de la bascule suivante ($Q_2{}^+ = \overline{Q_2}$);

– une instruction de test supplémentaire ($i = 13$) réalise la rétroaction $Z_6 = Q_4\,Q_2$; si le test est positif ($Q_4\,Q_2 = 1$), la bascule Q_2 du système câblé est remise à zéro, conformément au logigramme de la fig. 7a; la descente de Q_2 ($Q_2 = 1 \rightarrow 0$) assure la remise à zéro de Q_4 et le retour du diviseur dans l'état initial $Q_4, Q_2, Q_1 = 000$; dans le compteur programmé (fig. 7c: en trait fort), le même processus est assuré par la remise à zéro de l'état Q_2 lorsque $Q_4 = 1$ dans l'instruction de test $i = 13$;

– les deux instructions de test initiales ($i = 01$ et $i = 02$) détectent la variation de l'horloge H et provoquent un changement de l'état Q_4, Q_2, Q_1 à chaque montée de H ($H = 0 \rightarrow 1$).

Les douze instructions nécessitent 4 bits d'adresse, tandis que les quatre variables de test ($a \in \{H, Q_1, Q_2, Q_4\}$) et les trois registres de sortie (Q_1, Q_2 ou Q_4) requièrent 2 bits de codage a_1, a_0 (table V). Le format des instructions (en binaire) est alors le suivant:



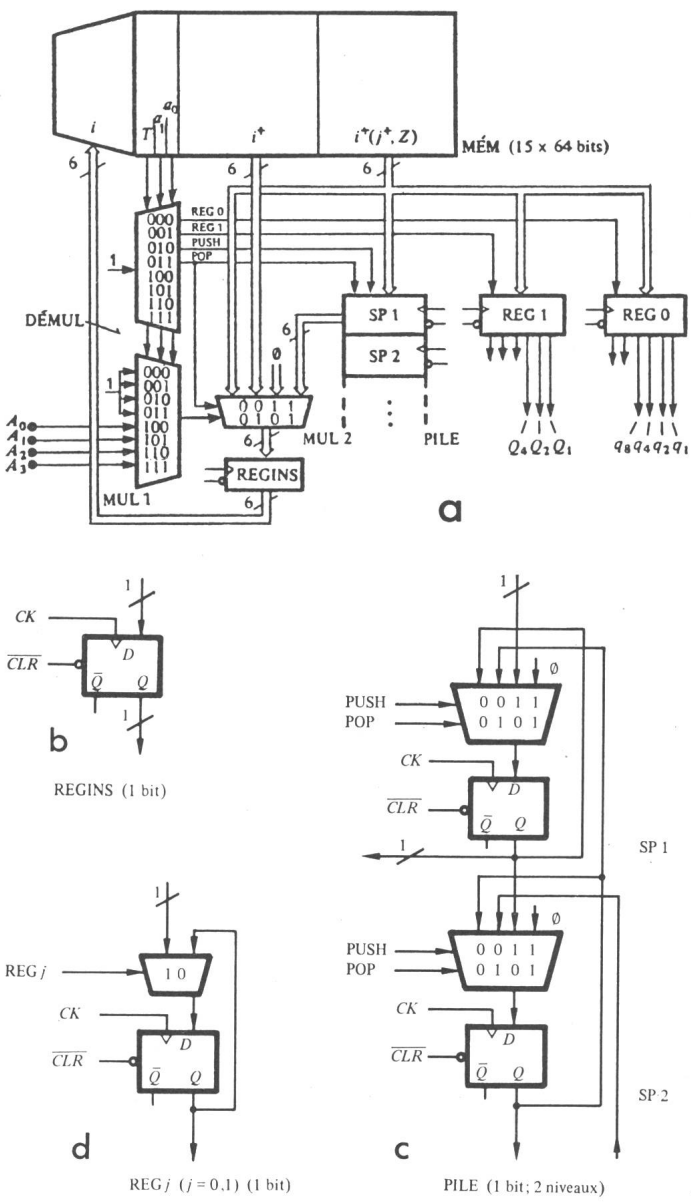


Fig. 6 Machine de décision binaire avec pile

a Schéma général; MÉM = mémoire; DÉMUL = démultiplexeur; MUL = multiplexeur

b Logigramme du registre d'instructions (REGINS)

c Logigramme de la pile (2 niveaux de sous-programmes SP1 et SP2)

d Logigramme des registres de sortie (REG1 et REG0)

dont on déduit le programme octal de la table VI.

Codage binaire des variables de test et des registres de sortie

Table V

T	$\underbrace{a}_{a_1\ a_0}$	Variable de test	Registre de sortie
1	0 0	H	–
	0 1	Q_1	–
	1 0	Q_2	–
	1 1	Q_4	–
0	0 0	–	REG 0 (Q_1)
	0 1	–	REG 1 (Q_2)
	1 0	–	REG 2 (Q_4)

Programme octal réalisant l'organigramme de la fig. 7c

Table VI

i	T	a	i^+ ($a = 1$) ou i^+	i^+ ($a = 0$) ou $Z = Qj$
00	0	2	01	00
01	1	0	01	02
02	1	0	03	02
03	1	1	04	10
04	0	0	05	00
05	1	2	06	11
06	0	1	07	00
07	1	3	00	12
10	0	0	01	01
11	0	1	13	01
12	0	2	01	01
13	1	3	06	01

En première approximation, c'est-à-dire en négligeant les instructions de la fonction de rétroaction, on dénombre $n = (3\ \lceil\log_2 p\rceil + 2)$ instructions et $\lceil\log_2 (1 + \lceil\log_2 p\rceil\ )\rceil$ bits de codage a_i. La capacité de la mémoire $C(p)$ vaut alors:

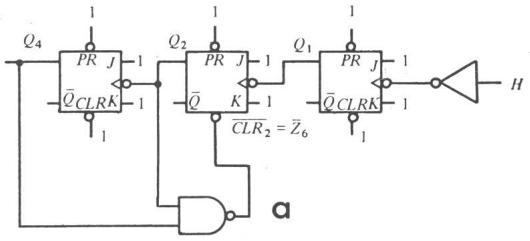
$$C(p) = nL \cong (3\ \lceil\log_2 p\rceil + 2)\ \left[1 + \lceil\log_2 (1 + + \lceil\log_2 p\rceil\ )\rceil + 2\ \lceil\log_2 (3\ \lceil\log_2 p\rceil + 2)\rceil\right]\ \text{[bit]} \quad (8)$$

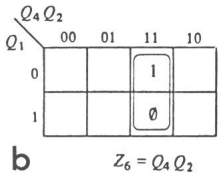
et devient dans l'exemple traité $C\,(6) \cong (3 \cdot 3 + 2)\ (1 + 2 + 2 \cdot 4) = 121$ bits.

En négligeant également les instructions de la fonction de rétroaction, la durée d'un cycle $d\,(p)$ vaut, dans le pire cas,

$$d(p) \cong 2 + 2\ \lceil\log_2 p\rceil \quad (9)$$

qui prend, dans l'exemple traité, la valeur $d\,(6) \cong 2 + 2 \cdot 3 = 8$.





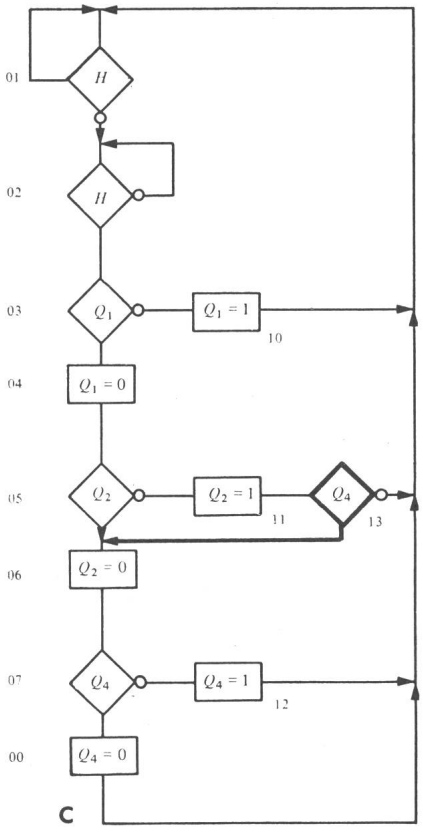


Fig. 7 Diviseur de fréquence par six
a Schéma du diviseur câblé
b Fonction de rétroaction
c Organigramme

Variantes pour la réalisation d'un compteur par soixante

Table VII

Cas	Type de programme	Capacité de la mémoire $C = nL$ bit	Durée d'un cycle d [1]	Facteur de mérite $C \cdot d$ bit	Nombre des niveaux de la pile
	Organigramme linéaire				
1	$p = 60$	$180 \cdot 17 = 3\,060$	3	9 180	–
	Arbre de décision				
2	$p = 60$	$121 \cdot 18 = 2\,178$	10	21 780	–
3	$p = 6 \cdot 10$	$32 \cdot 14 = 448$	11	4 928	–
4	$p = 6 \cdot 10$ (diviseur de fréquence)	$26 \cdot 14 = 364$	≈16	≈5 824	–
	Pile				
5	$p = 60$	$123 \cdot 17 = 2\,091$	5	10 455	1
6	$p = 6 \cdot 10$	$36 \cdot 15 = 540$	8	4 320	2
7	$p = (3 \cdot 2)\ (5 \cdot 2)$	$30 \cdot 14 = 420$	14	5 880	4

p mesure du compteur, n nombre d'instructions du programme, L largeur d'une instruction

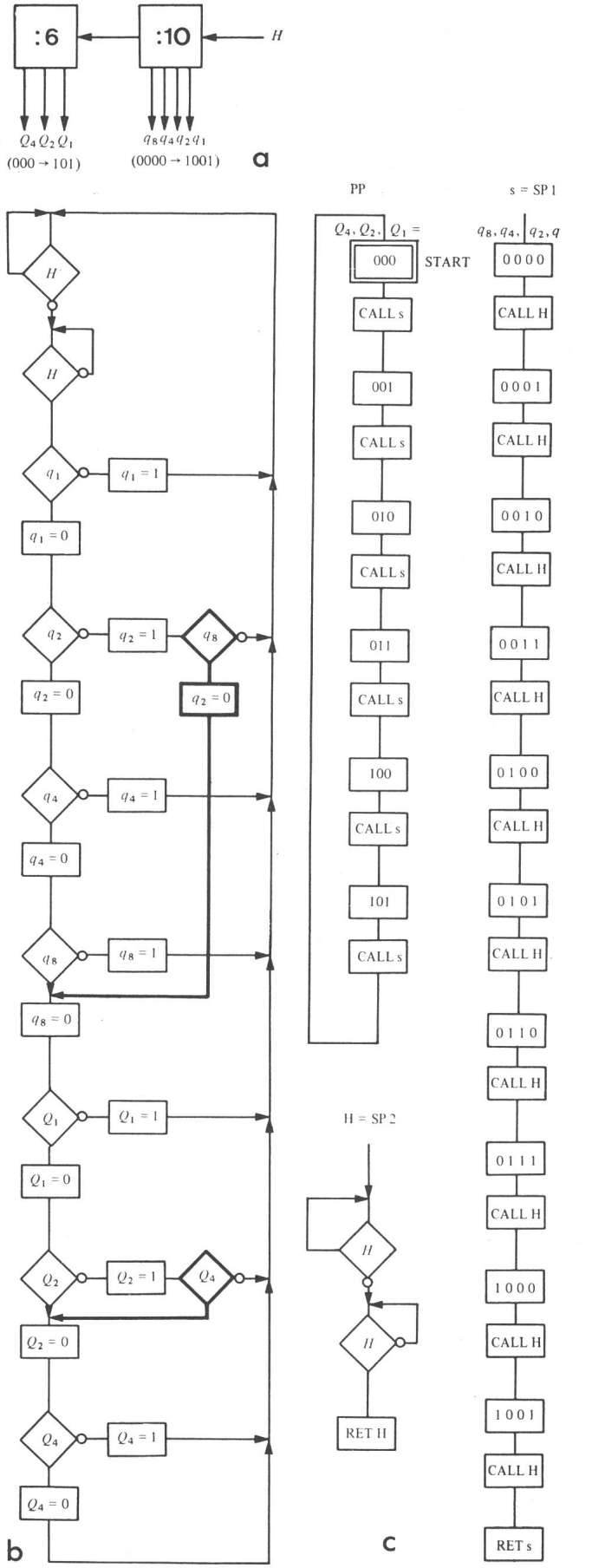


Fig. 8 Compteur par soixante

a Schéma

b Organigramme du diviseur de fréquence (cas nº 4) avec capacité C de la mémoire minimale

c Organigrammes du programme principal (PP) et de deux sous-programmes (SP 1 = s et SP 2 = H) réalisés par une pile (cas nº 6) avec un facteur de mérite $C \cdot d$ minimal
START: instruction initiale

5.3 Réalisation matérielle

La machine de décision binaire de la fig. 6 peut exécuter le programme de la table VI à condition

– d'assurer les connexions $A_0 = H$, $A_1 = Q_1$, $A_2 = Q_2$, $A_3 = Q_4$, nécessitées par le codage des variables de test (table V);

– de remplacer la pile par un registre REG 2, identique à REG 0 et REG 1, et commandé par la variable $PUSH = REG\,2$ qui est issue du démultiplexeur DEMUL; les sorties Q_4, Q_2 et Q_1 sont alors données par le bit d'extrême droite des registres REG 2, REG 1 et REG 0 (table V).

5.4 Commentaire

Un compteur câblé particulier, le diviseur de fréquence, a produit un programme d'une grande simplicité. Les états transitoires d'un tel diviseur câblé [3, pp. 119–121] seront également réalisés par le système programmé, étant donné les variations successives, et non pas simultanées, des trois sorties Q_1, Q_2 et Q_4 (fig. 7c). La séquence obtenue peut être décrite symboliquement par l'expression suivante:

$$Q_4, Q_2, Q_1 = 000 \rightarrow 001 \rightarrow (000) \rightarrow 010 \rightarrow 011 \rightarrow (010 \rightarrow 000) \rightarrow \rightarrow 100 \rightarrow 101 \rightarrow (100 \rightarrow 110 \rightarrow 100) \rightarrow 000 \qquad (10)$$

où les états transitoires sont représentés entre parenthèses.

5.5 Décomposition des compteurs

L'organigramme de la fig. 7c peut être considéré comme une décomposition de l'arbre de décision binaire de la fig. 4a, de même que le diviseur de fréquence câblé de la fig. 7a résulte de la décomposition en série d'un compteur par 8 ($8 = 2 \cdot 2 \cdot 2$), avec une rétroaction de 8 à 6. Cette même décomposition n'est pas applicable à l'organigramme linéaire (fig. 2), car elle entraînerait dans ce cas l'existence de processus parallèles. Par contre, des décompositions sans rétroaction (par ex.: $p = 6 = 3 \cdot 2$) sont réalisables à l'aide d'une pile et de plusieurs sous-programmes. Quelques applications de ces décompositions, étudiées théoriquement dans [3, pp. 148–151], sont illustrées dans les paragraphes 6.3 et 6.4.

6. Application: horloge digitale

6.1 Cahier des charges: compteur par soixante

Un tel compteur est essentiel dans toute horloge digitale pour le comptage et l'affichage des secondes d'une part, des minutes d'autre part. Le cahier des charges précise:

– la mesure ou rapport de division: $p = 60$;

– le code ou séquence des états (en décimal):
00→01→02→...→59→00.

En général, l'affichage décimal est effectué séparément pour les dizaines et pour les unités; on admet donc le code BCD (Binary Coded Decimal), qui nécessite 4 bits q_8, q_4, q_2, q_1 pour le comptage des unités et 3 bits Q_4, Q_2, Q_1 pour le comptage des dizaines (fig. 8a).

6.2 Variantes possibles

En se restreignant à l'existence d'une seule machine de décision binaire et d'un seul programme (on exclut donc les processus parallèles), on constate qu'un grand nombre d'organigrammes équivalents réalisent le cahier des charges. Ceux-ci sont obtenus à partir des trois modes principaux (organigramme linéaire, arbre et pile) et de leurs diverses décompositions, discutées au paragraphe 5.5. La table VII résume les

principales variantes, en indiquant dans chaque cas la capacité de la mémoire C, la durée d'un cycle d, ainsi qu'un *facteur de mérite* égal au produit Cd; on donne également le nombre des niveaux de la pile, égal au nombre des sous-programmes. Enfin, on a choisi d'illustrer deux des sept cas cités.

6.3 Compteur par soixante: diviseur de fréquence (Cas N° 4)

L'organigramme d'un tel compteur (fig. 8b) découle de la mise en série d'un compteur par dix (variables q_8, q_4, q_2, q_1) et d'un compteur par six, identique à celui de la fig. 7c (variables Q_4, Q_2, Q_1); chacun des compteurs partiels est un diviseur de fréquence muni d'une fonction de rétroaction adéquate ($Z_{10} = q_8\, q_2$ pour la rétroaction de 16 à 10, $Z_6 = Q_4\, Q_2$ pour la rétroaction de 8 à 6).

6.4 Compteur par soixante: pile et sous-programmes (Cas N° 6)

On distingue dans la fig. 8c le programme principal et les deux sous-programmes (s et H) qui mettent en évidence la clarté particulière de cette solution (programme structuré). La machine de décision binaire de la fig. 6, avec sa pile à deux niveaux (registres SP 1 et SP 2) et ses deux registres de sortie (REG 0 et REG 1), peut réaliser sans modification un tel organigramme.

7. Conclusion

Un compteur microprogrammé dont la mesure (ou rapport de division) est égale à p peut être réalisé:

– par un organigramme linéaire; dans ce cas, aucune décomposition n'est possible, mais la durée d'un cycle est minimale;

– par un arbre de décision binaire; celui-ci admet toujours des décompositions, dont la plus fine est constituée par le diviseur de fréquence (avec une éventuelle rétroaction). Si $p = p_1 \cdot p_2 \cdot \ldots \cdot p_q$, on constate que le nombre des instructions n est approximativement proportionnel à la somme des mesures partielles ($p_1 + p_2 + \ldots + p_q$); la décomposition la plus fine (le diviseur de fréquence) produit donc le nombre n minimal;

– par une pile; celle-ci admet des décompositions si la mesure $p = p_1 \cdot p_2 \cdot \ldots \cdot p_q$ est un produit de nombres entiers (aucune rétroaction n'est possible). Comme dans le cas de l'arbre, le nombre des instructions n varie comme la somme des mesures partielles ($p_1 + p_2 + \ldots + p_q$).

La décomposition d'un compteur entraîne donc une diminution de la capacité de la mémoire, mais elle implique, en contrepartie, une augmentation de la durée d'un cycle et, le cas échéant, une croissance de la taille de la pile (un registre pour chaque sous-programme).

Les travaux relatés dans cette publication découlent d'une étude sur la conception d'un microprocesseur horloger [5]; cette étude, financée en partie par la Commission pour l'encouragement des recherches scientifiques (projet No 988), réunit notamment le Centre électronique horloger (Neuchâtel), l'Institut de microtechnique de l'Université (Neuchâtel), la Chaire de systèmes logiques et le Laboratoire de calculatrices digitales de l'EPF Lausanne, ainsi que les principales entreprises horlogères suisses.

Bibliographie

[1] *D. Mange:* Arbres de décision pour systèmes logiques câblés ou programmés. Bull. ASE/UCS 69(1978)22, p. 1238...1243.
[2] *A.D. Hearn:* Some words about program structure. Byte 3(1978)9, p. 68...76.
[3] *D. Mange:* Analyse et synthèse des systèmes logiques. Traité d'électricité. Vol. V. Saint-Saphorin, Editions Georgi, 1978.
[4] *E. Cerny, D. Mange* and *E. Sanchez:* Synthesis of minimal binary decision trees. IEEE Trans. C 28(1979)7 (à paraître).
[5] *E. Sanchez* et *A. Stauffer:* Horloge microprogrammée. 10e Congrès International de Chronométrie, Genève, septembre 1979 (à paraître).
[6] *D.E. Knuth:* The art of computer programming. Vol. I: Fundamental algorithms. Reading/Massachusetts, Addison-Wesley, 1969.

Adresse de l'auteur

Daniel Mange, Professeur EPFL, Chaire de systèmes logiques, 16, chemin de Bellerive, 1007 Lausanne.