VITO ASTA

LA SYNTHESE NUMERIQUE EN TEMPS RÉEL

N° 103

JUIN 1980

GAM

BULLETIN DU GROUPE d'ACOUSTIQUE MUSICALE
UNIVERSITE PARIS VI - TOUR 66 - 4 PLACE JUSSIEU - PARIS 5°

Université Paris VI
Groupe d'Acoustique Musicale
Laboratoire d'Acoustique
4 Place Jussieu-Tour 66
75005 - PARIS

Paris le 5 Décembre 1985

103ème Bulletin du GAM

LA SYNTHESE NUMERIQUE EN TEMPS REEL

par VITO ASTA

Etaient présents :

E.LEIPP (Dir. de Recherches au CNRS) ; M. CASTELLENGO; J.KERGOMARD (CNRS);M.TESTEMALE(Informaticien); M.TARRIERE(Etud.); M.CORDEAU(Prof;) M.COMBASTET; Mme ALLOUIS(architecte); M.FERRON(luthier);M.LIJOUR(LNE) M.etMme MULLETIN; M.WU(M.A.); S.CHAINTREUIL(Etud.);L.ETIENNE(Prof.); E.CHUILON(Etud.); R.CAUSSE(IRCAM); J.DARNIS(CNRS); J.HAMON(Etud.); J.Y.BERNIER(Etud.); Mme KADRI(Médecin);S.HUE(Prof.); Mlle BAZANTE; G.SABAN; Ch.BESNAINOU(CNRS).

Excusés :

M.SIESTRUNCK; M.BIEZUNSKI; M.AROM; M.J.CHAILLEY; M.A.BARON; M.MOIROUD; M.BAERD; Mme STRAUS; Ph.BOURGOIN; J.M.FONTAINE; M.CARRE; M.PORTEOUS; M.JOUHANEAU; J.LEGUY; P.FRANCOIS; M.GEUENS; J.L.MASSON; Ph.LAROCHE; M.CATRICE; M. LEHMAN; Mme BRAN-RICCI; M. GUIRAUD.

* * *
*

TABLE DE MATIERES

La lecture de cet exposé est de difficulté croissante. Le début devrait être accessible à des personnes n'ayant aucune connaissance de base sur les techniques ni sur les principes fondamentaux de la synthèse du son. Progressivement, il est demandé au lecteur un certain "background" mathématique, puis électronique et informatique, pour suivre le discours jusqu'au bout.

Les références aux notes sont marquées par des numéros entre crochets []. Aucune de ces notes n'est essentielle à la compréhension du texte à un premier niveau. Au contraire, toutes les notes sont écrites en général dans un langage plus spécialisé, et ne devraient probablement être consultées qu'en deuxième lecture; c'est pour cette raison que les notes sont groupées à la fin du texte.

1. Introduction

Le problème de la composition musicale se pose aujourd'hui, pour le musicien d'avantgarde, en des termes nettement différents par rapport aux canons traditionnels. Le musicien désire pouvoir contrôler directement tout aspect du son: temps, fréquence, amplitude, spectre, localisation dans l'espace; tout cela doit être l'objet d'un choix qui désormais n'est plus opéré dans un ensemble discret et limité de valeurs, comme c'était le cas dans le passé, mais dans un intervalle continu de variation de chacun des paramètres. Le compositeur désire pouvoir opérer, pour ainsi dire, une "microchirurgie du son": autrement dit, il désire, plutôt que composer avec les sons, composer le son lui-même, en explorant ses plus intimes aspects. On peut, naturellement, être d'accord ou non avec une telle façon de concevoir l'activité compositionnelle, et nous n'approfondirons pas le problème davantage; en tout cas, on peut au moins observer que l'activité de tout compositeur a toujours été reliée étroitement, dans l'histoire, à une confrontation constante avec les moyens que l'état de la technique lui mettait à disposition; et la situation de notre temps ne fait pas exception.

De là est née cette exigeance, ressentie de plus en plus fortement ces dernières années, de pouvoir disposer de moyens dont la précision soit toujours plus poussée: après les expériences historiques de Pierre SCHAEFFER **[1952]** dans les années '50, qui ont eu au moins la mérite de bien mettre en évidence le problème de devoir trouver quelque chose "d'autre", les travaux de Karlheinz Stockhausen **[PRIEBERG, F.K., 1960]** dans le studio de la West-Deutscher Rundfunk à Cologne marquèrent le début de l'utilisation de systèmes électroniques pour la production directe de la musique: le son était engendré directement par des oscillateurs, qu'on pouvait moduler entre eux à l'aide des classiques modulateurs en anneau, originalement utilisés dans les transmissions radio, ou bien filtrer; le résultat était enregistré sur bande magnétique, et après beaucoup de travail aux ciseaux on pouvait recoudre toute une pièce. Vers la moitié des années '60, Robert Moog, Donald Buchla et Paul Ketoff **[MOOG, R.A., 1965; CHADABE, J., 1972]** introduirent presque simultanément la technique du contrôle en tension (Voltage Control), qui représentait déjà un progrès formidable par rapport à la situation précédente: les vieux oscillateurs à contrôle manuel sont maintenant remplacés par des oscillateurs contrôlés en tension (Voltage Controlled Oscillator ou VCO), c'est à dire dont la fréquence est reglée par une tension de commande, qui à son tour peut être engendrée par la sortie d'un autre composant du système; de même, les filtres ont eux aussi des entrées de contrôle en tension qui règlent la ou les fréquences de coupure, qu'on peut donc faire varier à des vitesses jusqu'alors impossibles (Voltage Controlled Filter ou VCF); le troisième élément de base est l'amplificateur contrôlé en tension (Voltage Controlled Amplifier ou VCA), qui permet de contrôler l'évolution globale de l'amplitude d'un signal donné [1], que ce soit le résultat global ou simplement une variable intermédiaire. Chacun

de ces modules peut être connecté à n'importe quel autre, en reliant les entrées et les sorties par des petits câbles, pour implanter ainsi le processus de synthèse voulu.

L'extrême modularité de ces unités de base permit rapidement leur intégration en des systèmes compacts, plus ou moins portatifs, souvent vendus avec un clavier d'orgue pour la génération des tensions de contrôle des fréquences des oscillateurs, qu'on a nommé synthétiseurs (analogiques); bien que le clavier permette de s'en servir comme un instrument de musique, sur le vif, il faut plutôt considérer les synthétiseurs comme des petits laboratoires transportables d'alchimie du son. Il faut reconnaître que ces dispositifs, bien qu'extrêmement imprécis et limitatifs quant aux techniques de synthèse utilisables [2], ont eu, par leur diffusion, une importance capitale sur le développement de la musique électronique dans les années '60 et '70.

Pour remédier aux défauts qu'on vient de préciser, enfin, et à d'autres qu'on détaillera au cours de cet exposé, déjà vers la fin des années soixante un certain nombre de musiciens et chercheurs plus sensibles au problème voyait dans l'ordinateur la seule réponse satisfaisante à leurs nécessités **[MATHEWS, M.V., et al., 1969; RISSET, J.C., 1965 et 1969]**. Dès lors, les méthodes de synthèse musicale par ordinateur se sont étendues très rapidement, jusqu'à comprendre aujourd'hui en pratique toutes les techniques de traitement numérique du signal, dont la Computer Music, comme on appelle désormais ce nouveau tournant de la musique électronique, est à présent un terrain d'application très fertile **[MOORER, J.A., 1977; ASTA, V., 1978]**.

2. Musique et ordinateurs

Si on peut facilement arriver à voir comment obtenir des sons par les techniques de contrôle en tension, la même démarche, dans le cas des ordinateurs, est certainement moins évidente **[MOORER, J.A.,1978 b]**.

Il est bien connu que les ordinateurs ne travaillent que sur des nombres, donc le premier problème qu'il faut résoudre est celui de la conversion, ou mieux de la représentation, d'un son (sous forme de tension électrique, après transduction par un microphone) par une suite de nombres. Cela peut être obtenu en effectuant une mesure de la forme d'onde électrique à un instant donné: le résultat est alors un nombre, qui indique par exemple combien de volts mesurait le signal électrique à l'instant de la mesure. Si on effectue des mesures répétées du signal, à une cadence suffisamment rapide, l'ensemble de ces nombres (ou échantillons) fournira une description suffisamment détaillée du signal analogique (le signal sous forme électrique est dit "analogique" parce-qu'il fournit une représentation du signal acoustique dont l'évolution dans le temps est analogue à celle de l'original; ce terme est souvent utilisé en opposition à

"numérique" ou "digital" - de digit, chiffre -, qu'on utilise justement pour les représentations au moyen d'une série de nombres). Un célèbre théorème, dit théorème de Shannon ou de l'échantillonnage **[THOMAS, J.B., 1969]**, précise à quel taux il faut mesurer le signal pour en avoir une représentation précise (dans le sens qu'il soit possible de reconstituer exactement, à partir de la suite de nombres obtenus et par un processus inverse de celui qu'on vient de décrire, le signal de départ). Ce taux (ou fréquence d'échantillonnage) dépend de la largeur de bande - c'est à dire du contenu en fréquences - du signal, comme il est intuitif de le comprendre: plus un signal est pauvre en fréquences aiguës, plus il varie lentement dans le temps, et donc les mesures nécessaires peuvent être plus espacées. Précisément, si F_{max} est la fréquence maximum (en Hz) contenue dans le signal, le taux minimum qui assure une bonne représentation est de $2 \times F_{max}$ échantillons par seconde. Par exemple, pour reconstituer fidèlement un bon signal de musique, dont les fréquences peuvent arriver à environ 16000 Hz, il faut prélever au moins 32000 échantillons par seconde, soit un échantillon tous les 30 usec (micro-secondes) environ.

Un dispositif électronique capable d'effectuer la conversion d'un signal électrique en une suite de nombres est appelé convertisseur analogique-numérique (analog to digital converter ou ADC); le dispositif qui réalise la fonction inverse est appelé convertisseur numérique-analogique (digital to analog converter ou DAC). Il existe aujourd'hui dans le commerce plusieurs types de DAC et de ADC, avec différentes vitesses de fonctionnement selon les applications. Un deuxième paramètre très important pour ces dispositifs, à part la vitesse, est le nombre de chiffres binaires ou bits [3] qu'ils produisent comme résultat de la conversion (pour les ADC) ou qu'ils acceptent en entrée pour effectuer la conversion (pour les DAC): il est évident que ce nombre détermine la précision avec laquelle la conversion est effectuée, et donc la qualité du signal converti. Pour des applications téléphoniques, par exemple, 8 bits suffisent; pour la musique, on utilise normalement entre 12 et 16 bits.

Si on dispose donc d'un ordinateur équipé au moins d'un convertisseur numérique-analogique, il est clair que le problème de la synthèse des sons se réduit maintenant à celui de faire un bon programme, qui fasse sortir à l'ordinateur "les bons numéros" (ceux qui correspondent au résultat sonore qu'on cherche à obtenir). L'ordinateur permet donc de remplacer tous les modules de base du studio analogique classique par autant de "dispositifs logiciels": chaque élément devient un sous-programme, qui simule numériquement le bloc analogique correspondant; on aura donc des sous-programmes oscillateurs, des sous-programmes filtres, et ainsi de suite. Les interconnexions entre ces blocs se réalisent maintenant par des simples passages d'arguments entre un sous-programme et un autre.

En utilisant les différents blocs qu'un programme de synthèse donné met à sa disposition, le musicien prépare donc, en

spécifiant les interconnexions, ce que, par évidente analogie, on appelle les "instruments", c'est à dire les unités de génération du son, aux paramètres desquels il attribue ensuite des séquences de valeurs d'entrée qui constituent la "partition".

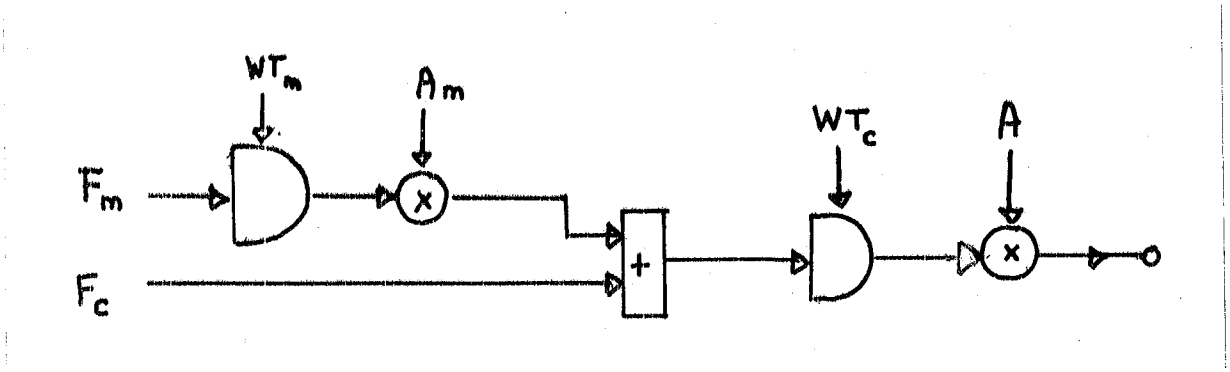

fig. 1 - simple instrument à modulation de fréquence

Considérons, par exemple, le schéma de fig. 1, représentant un simple instrument à modulation de fréquence: on peut voir qu'il est constitué d'un premier oscillateur avec fréquence F_m et forme d'onde WT_m, d'un multiplicateur qui multiplie la sortie de l'oscillateur par A_m, d'un additionneur qui ajoute F_c au résultat précédent, d'un deuxième oscillateur avec fréquence contrôlée par la sortie de l'additionneur - sa fréquence sera donc F_c quand A_m est nul - et forme d'onde WT_c, et d'un multiplicateur final qui multiplie la sortie du deuxième oscillateur par A; les paramètres de contrôle de cet instrument sont donc au nombre de 6, à savoir: F_m, WT_m, A_m, F_c, WT_c et A. Une réalisation de l'instrument sur un synthétiseur analogique nécessite l'emploi de deux VCO (pour les oscillateurs), deux VCA (pour les multiplicateurs), et un mixer à deux entrées (pour l'additionneur); ces 5 unités sont à interconnecter physiquement, par des fils, de la même façon que sur la figure. Pour "jouer" l'instrument il faut donner des valeurs ou des suites de valeurs aux paramètres de contrôle, par exemple manuellement. Une réalisation sur ordinateur du même instrument appelle deux fois une routine "oscillateur", deux fois une routine "multiplicateur" et une fois une routine "additionneur"; cette dernière, par exemple, reçoit deux paramètres d'entrée, qui seront F_c (paramètre externe) et le paramètre en sortie de la routine multiplicateur (premier appel), et donne à la sortie un paramètre, qui sera utilisé comme entrée de fréquence pour la routine oscillateur (deuxième appel); on procède ainsi de suite pour la réalisation de toutes les interconnexions de l'instrument. La partition de cet instrument consistera donc d'une série de valeurs pour les 6 paramètres de contrôle.

Plus précisément, on peut distinguer 4 phases fondamentales dans le processus de synthèse du son par ordinateur:

- l'utilisateur définit les instruments, en spécifiant les interconnexions entre les blocs (sous-programmes) à disposition;

- l'utilisateur définit la partition, en spécifiant les séquences de valeurs à assigner aux paramètres globaux des instruments définis au pas 1, et les instants auxquels ces assignations doivent être faites;

- l'ordinateur, sur la base des informations reçues dans les pas 1 et 2, effectue le calcul, échantillon par échantillon, du ou des signaux de sortie du processus ainsi défini; au fur et à mesure que les résultats sont calculés, ils sont transférés de la mémoire centrale à un dispositif à grande capacité de mémoire (comme un disque ou une bande magnétique); cette phase peut demander un temps de calcul total très long;

- l'ordinateur démarre un programme d'exécution de la pièce mémorisée durant le pas 3, qui recharge en mémoire centrale, par blocs, le contenu de la bande ou du disque et envoie à un DAC les échantillons, à la bonne vitesse, pour la conversion en signal électrique continu et enfin en son (les DACs peuvent être plusieurs, en cas de différents signaux synthétisés et à exécuter en même temps).

Les avantages d'un tel système sont bien évidents; entre autres, on peut citer les suivants:

- une flexibilité du système nettement supérieure: il est toujours plus facile de changer, par exemple, les caractéristiques d'un oscillateur logiciel, en réecrivant quelques lignes de programme, que de changer un oscillateur matériel, qui est physiquement figé;

- une stabilité des paramètres incomparablement meilleure: le problème, bien connu de ceux qui ont eu quelques contacts avec des oscillateurs analogiques, de la dérive de la fréquence effective avec la température du dispositif, n'a aucun équivalent dans les systèmes numériques;

- une complexité virtuellement illimitée soit pour ce qui concerne le nombre d'unités (un programme oscillateur peut simuler autant d'unités qu'on veut: il suffit de l'appeler plusieurs fois de suite, avec des paramètres d'entrée différents), soit pour la longueur des séquences à exécuter (les synthétiseurs analogiques les plus sophistiqués sont équipés de dispositifs, dits séquenceurs, qui peuvent sortir en boucle une série d'une trentaine de valeurs au maximum; le reste, il faut le faire à la main), soit pour le nombre de voix, soit, enfin, pour les formes d'onde possibles de

chaque oscillateur (qu'on peut programmer à son gré, alors que la quasi-totalité des synthétiseurs, pour des raisons liées aux caractéristiques physiques des oscillateurs analogiques, ne possède que 4 formes d'onde possibles: sinusoïde, onde carrée, onde triangulaire et dent de scie).

Après avoir énuméré tous ces avantages, il faut cependant noter l'existence d'au moins un point négatif: dans le processus de synthèse proprement dit (la troisième phase décrite précédemment), le compositeur reste complètement en dehors du cycle: une fois définis les instruments et la partition pour une séquence donnée, l'ordinateur travaille longtemps en solitaire; ce n'est que dans un deuxième moment que le compositeur peut écouter le résultat. Si, comme il arrive la plupart des cas, il n'est pas satisfait des sons obtenus, il doit changer les partitions (et éventuellement les instruments) et recommencer toute la procédure. Tout cela peut demander un temps prohibitif si le temps de calcul de l'ordinateur (qui naturellement dépend de la complexité des instruments introduits et de la longueur des partitions assignées) est très long.

Si on définit une échelle des temps ET comme suit:

$$ET = \frac{\text{temps de calcul}}{\text{durée totale des sons calculés}}$$

on peut dire que les valeurs de ET qu'on rencontre en pratique sont, dans les cas les plus compliqués, de 50 à 100 environ (ce qui signifie qu'une minute de musique peut demander jusqu'à 100 minutes de calcul). Avec les instruments traditionnels et les synthétiseurs analogiques [4], ce genre de problème ne se posait pas du tout: le son sortait en même temps que les processus pour l'engendrer se déroulaient, autrement dit on avait une valeur de ET égale à 1; dans ce cas, on dit qu'il s'agit d'un instrument fonctionnant "en temps réel".

Si donc on arrive à obtenir, d'un ordinateur, une échelle des temps égale (ou inférieure) à 1, on a un instrument numérique en temps réel; les avantages d'un tel système sont nombreux et très importants; on peut rappeler, entre autres, les suivants:

- les temps de travail sont, en général, plus rapides;
- le temps de calcul de l'ordinateur (et donc le coût de chaque passage du programme, si l'ordinateur n'est pas acheté) est moindre;
- le système est accessible à un plus grand nombre d'utilisateurs, étant donné que chacun d'eux occupe moins de temps de travail disponible;
- l'utilisateur peut communiquer avec l'ordinateur au moment même où la synthèse se déroule, au moyen d'un ou plusieurs dispositifs de contrôle gestuel (comme des potentiomètres, des pédales ou autre) [5]; il peut ainsi déterminer

l'évolution de certains paramètres des instruments en jeu en même temps qu'il écoute le résultat, sur le signal acoustique final, produit par ces variations. Cela permet évidemment de déterminer les bonnes valeurs des paramètres en des conditions extrêmement favorables; de l'autre côté, le musicien peut "jouer" l'ordinateur, en exécutant sur le vif les séquences de certains paramètres.

Pour minimiser l'échelle des temps, il n'y a évidemment que deux possibilités: soit réduire la complexité des instruments, soit faire des ordinateurs plus rapides. Malheureusement, les ordinateurs dont on dispose à l'heure actuelle [6] sont trop lents pour pouvoir réaliser une synthèse en temps réel avec des instruments de complexité acceptable; cela est dû essentiellement à deux causes: la première est qu'un ordinateur, qui par définition est une machine tout à fait générale, doit, à tout moment, prendre un nombre relativement élevé de décisions, avant d'exécuter quelque chose: et plus un ordinateur est versatile, plus son jeu d'instructions est vaste, plus il lui faudra prendre de décisions avant de passer à l'exécution d'une instruction. La deuxième raison est que, toujours à cause de la généralité de la machine, le nombre de bits des nombres sur lesquels un ordinateur travaille ne correspond pas exactement, en général, à la précision strictement nécessaire pour une application donnée: trop de bits signifie qu'une partie des calculs est effectuée inutilement, alors que trop peu signifie qu'il faut faire deux opérations pour une (on regroupe deux numéros pour en représenter un seul).

La solution au problème du temps réel est alors évidente, au moins dans son principe: il faut construire des ordinateurs spécialisés pour l'application envisagée (dans notre cas, la synthèse de signaux sonores à partir de certains algorithmes), c'est à dire avec la précision (voir nombre de bits des registres de travail et de toute l'architecture interne du système) exactement nécessaire, et avec un jeu d'instructions extrêmement reduit, et parfois nul: dans ce cas, le micro-programme est directement câblé dans le système, et est donc figé une fois pour toutes: ces ordinateurs (si on peut encore les appeler ainsi) ne savent faire qu'une chose, mais ils la font très rapidement. C'est à cette catégorie de systèmes, apparus dans la deuxième moitié des années '70 et dont la venue a révolutionné encore une fois le panorama de la musique contemporaine, qu'on donne le nom de synthétiseurs digitaux en temps réel.

3. Réalisation des oscillateurs

Avant de parler spécifiquement des techniques de synthèse qu'on peut utiliser, il est opportun de détailler le fonctionnement et la structure interne du bloc oscillateur **[MATHEWS, M.V., et al., 1969]**, soit dans une mise en oeuvre logicielle (pour un programme général de synthèse), soit dans une réalisation matérielle (pour un synthétiseur digital). Ce bloc

est évidemment le plus important de tous, étant donné que c'est par lui qu'il faut nécessairement passer pour produire un signal initial, que les autres modules pourront éventuellement modifier avant de sortir le résultat final.

Supposons, à titre d'exemple, que nous voulons réaliser par programme un oscillateur sinusoïdal, exprimable par la formule

$$(1) \qquad X(n) = \sin(\omega_o \; n \; T + \varphi_o)$$

où:

- X(n) est le signal qu'on veut obtenir par l'oscillateur, au temps nT (T est la période d'échantillonnage, n est un entier);
- $\omega_o = 2 \pi f_o$ exprime la fréquence de l'oscillateur;
- φ_o est l'éventuelle phase initiale de l'oscillateur;
- L'amplitude de l'oscillateur est fixe et égale à 1 (elle peut être modifiée par un bloc multiplicateur connecté à la sortie de l'oscillateur).

Si on réalisait cet oscillateur par calcul direct de la formule (1), à chaque échantillon l'ordinateur devrait effectuer un calcul de la fonction sinus, ce qui demande plusieurs additions et multiplications. Cela peut signifier un travail pour l'ordinateur, et donc un temps de calcul, absolument inacceptable dès que la durée du signal à synthétiser dépasse les quelques secondes, surtout si l'instrument qu'on utilise requiert plusieurs oscillateurs (ce qui est souvant le cas). D'autre part, la fonction sinus est toujours la même, et tous les $2 \pi / \omega_o \; T$ échantillons les valeurs se répètent inchangées. On peut alors mémoriser au préalable ces échantillons une fois pour toutes, dans une zone de mémoire de l'ordinateur, et adresser ensuite cette mémoire en séquence, moyennant un incrément, à chaque échantillon, d'un compteur modulo $2 \pi / \omega_o \; T$ (qu'on réalisera par une variable du programme). De cette façon, au lieu du calcul de la fonction sinus il suffit de faire une addition et une lecture de mémoire.

Cette méthode, telle qu'on l'a énoncée jusque là, n'est valable que pour la fréquence f_o et pour la phase φ_o; pour réaliser un oscillateur de fréquence et phase quelconques on étend la technique comme suit: on mémorise dans un bloc de mémoire la forme d'onde; soit LW la longueur de ce bloc, c'est-à-dire le nombre de nombres qu'il contient; on mémorise dans une variable (représentant la phase instantanée de l'oscillateur) la phase initiale et on l'incrémente à chaque fois d'un pas opportun, dépendant de la fréquence qu'on veut obtenir; le résultat de cette addition, modulo LW, détermine l'adresse (contenue dans le bloc de mémoire) de l'échantillon à sortir. Si l'incrément est égal à 1, la période du signal obtenu sera LW

fois la période d'échantillonnage T (puisque après LW lectures de la table on recommence à lire les mêmes nombres); si l'incrément double, la période se réduit à la moitié (il suffit maintenant de la moitié de lectures pour arriver à la fin de la table et recommencer), donc la fréquence double aussi; en général donc, pour un incrément H, la fréquence qu'on obtient sera donnée par la formule

$$(2) \qquad F=\frac{H}{LW \times T}$$

qu'on peut réécrire

$$(2') \qquad F=\frac{LW \times F}{F_c}$$

où F_c est la fréquence d'échantillonnage correspondant à T.

La formule (2') permet de calculer l'incrément nécessaire pour obtenir une fréquence donnée; il est facile de se convaincre que, en général, H est un nombre réel, avec une partie fractionnaire qui est très importante pour la précision des fréquences. Par exemple, si on échantillonne à 32 kHz et qu'on utilise des tables de $2^{10} = 1024$ nombres, pour avoir une fréquence de 100 Hz il faut utiliser un incrément égal à 3.2; une valeur de 3 donnerait une fréquence de 93.75 Hz, ce qui représente une erreur de 6.25%, soit plus d'un demi ton.

Après avoir effectué l'addition de l'incrément à la phase, celle-ci est convertie en un nombre entier pour pouvoir en obtenir une adresse (l'adresse de l'échantillon). Cela se fait, par exemple, par simple tronquage de la phase: si par exemple l'incrément vaut 6.4, les résultats des accumulations de l'incrément et les adresses des échantillons à prélever dans le tableau sont indiqués dans la table 1.

Table 1

numéro d'échantillon	phase instantanée	adresse de l'échantillon
1	6.4	6
2	12.8	12
3	19.2	19
4	25.6	25
5	32.0	32
6	38.4	38
7	44.8	44

Comme on le voit, la suite d'adresses est apparemment irrégulière, mais elle se compense de façon interne (grâce à l'accumulation, dans la phase, des parties fractionnaires de l'incrément) pour reproduire exactement la fréquence voulue. Tout cela permet d'avoir une grande précision dans la définition des fréquences, qui ne serait pas possible autrement [7]. L'Appendice I montre, à titre d'exemple, une réalisation possible de la routine oscillateur en langage FORTRAN.

Cette technique, aujourd'hui employée dans la totalité des programmes de synthèse, est connue sous le nom de "table lookup" (consultation de tableau).

Passons maintenant au problème de la mise en oeuvre des oscillateurs sous forme de circuits numériques concrets; on peut dire tout de suite que la technique utilisée est essentiellement la même: on aura ici (voir fig. 2) deux registres de mémoire, un qui mémorise l'incrément correspondant à la fréquence voulue (et qui est donc changé exclusivement par l'ordinateur qui gère le synthétiseur), l'autre qui contient la valeur de la phase instantanée, et qui donc est modifié tout le temps par le synthétiseur lui-même: une fois par période d'échantillonnage, les contenus des deux registres sont additionnés entre eux; le résultat de l'addition est mémorisé à nouveau dans le registre de la phase (pour le prochain cycle), et en même temps une partie des bits de ce résultat (les plus significatifs) va adresser un circuit de mémoire, contenant plusieurs emplacements (appelé couramment RAM: random access memory) que l'ordinateur pilote aura chargé précédemment avec la forme d'onde voulue. Ce circuit sort, suivant l'adressage, l'échantillon actuel de l'oscillateur. Comme on le voit, presque rien n'a changé, au moins d'un point de vue fonctionnel, entre cette situation et la précédente: la conversion de la phase réelle en adresse entière correspond ici au prélèvement des bits les plus significatifs du résultat de la somme (on effectue donc un tronquage; dans l'exemple de la figure, l'accumulation de la phase est effectuée sur 24 bits, mais seuls les 10 premiers sont utilisés pour adresser la mémoire (qui contient 2^{10} = 1024 échantillons). Le registre de phase peut donc être considéré comme composé de 10 bits de partie entière et 14 bits à partie fractionnaire. D'un autre côté, l'adressage circulaire est assuré automatiquement par la longueur finie des registres: une fois arrivé au maximum de sa valeur, à l'incrément successif le registre de la phase subit un débordement de sa capacité, perd le bit le plus significatif et recommence donc à zéro. Si, pour fixer les idées, le registre en question avait une capacité de 3 chiffres décimaux, il pourrait compter de 000 à 999. Une fois arrivé au maximum, un incrément de 2 ne l'amènerait pas à contenir 1001 (qui n'est pas représentable avec trois chiffres), mais il perdrait le 1 des milliers et se retrouverait à la configuration 001.

La précision en fréquence de ces oscillateurs est déterminée par la longueur en bits des registres de phase et d'incrément: le plus gros incrément qu'on peut écrire correspond à la fréquence maximum qu'on peut avoir, soit $F_{max} = F_c / 2$ (pour le théorème de Shannon). Si m est le nombre de bits de ces registres, ce nombre maximum est 2^{m-1}, étant donné qu'un bit est pris pour déterminer le signe du nombre; donc les fréquences de 0 à F_{max} sont divisées en 2^{m-1} intervalles égaux. La valeur de cet intervalle est la fréquence la plus petite qu'on peut avoir, et aussi l'écart minimum en fréquence qu'on peut réaliser; il est donc donné par

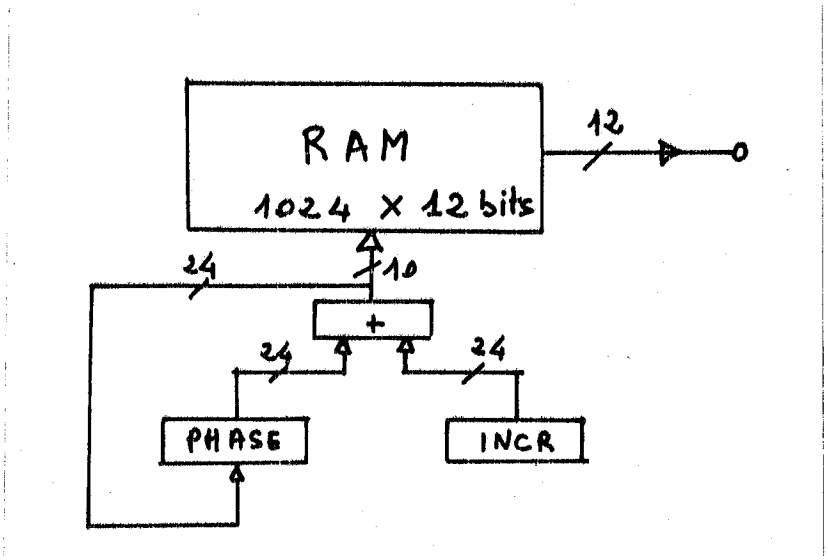

fig. 2 - schéma de réalisation d'un oscillateur numérique par table look-up. les détails sur les longueurs de mot et sur la taille mémoire sont donnés à titre indicatif

$$df = \frac{F_c / 2}{2^{m-1}} \qquad (3)$$

Par exemple, si on échantillonne à 32 kHz et qu'on utilise des registres de phase et d'incrément à 24 bits, la précision en fréquence vaut

$$df = 16000 / 2^{23} = 1.907 \times 10^{-3} \text{ Hz}$$

soit environ 2 millièmes de Hertz.

4. Les techniques de synthèse

Dans les paragraphes qui suivent on cherchera donc à passer brièvement en revue les principales techniques de synthèse du son à la disposition du compositeur. Ces techniques sont en général réalisables soit par des programmes généraux de synthèse [8], soit par des synthétiseurs digitaux, soit enfin, évidemment, par tout autre programme "ad hoc" pour une composition particulière.

On peut distinguer, à l'heure actuelle, quatre types fondamentaux de synthèse, qui sont: la synthèse additive, la synthèse soustractive, la synthèse par modulation de fréquence (FM) et la synthèse par distorsion non linéaire (ou waveshaping) [9]; certains de ces types, comme on le verra, ont aussi des applications comme techniques de synthèse à partir de l'analyse (d'un son donné), c'est à dire qu'il est possible d'obtenir les paramètres de contrôle de l'instrument, de façon plus ou moins

automatisée, sur la base de l'analyse des caractéristiques d'un ou plusieurs sons naturels (comme des instruments traditionnels ou la voix humaine). Il est opportun de préciser, à ce propos, qu'en général il n'est d'aucun intérêt en soi, du point de vue strictement musical, de chercher à reproduire fidèlement le son des instruments déjà existant, puisque le son d'une vraie clarinette sera toujours plus beau et convaincant que celui d'une clarinette synthétique, aussi sophistiqué que soit le programme ou le matériel qui le produit. La raison pour laquelle on cherche à bien savoir simuler les instruments traditionnels est double: on cherche à acquérir une plus grande connaissance des timbres naturels, soit du point de vue strictement physique (c'est l'objet de l'acoustique musicale classique) soit du point de vue psycho-acoustique (en particulier, déterminer l'influence, dans la perception du timbre, des évolutions de l'attaque et de l'extinction du son, du spectre, et d'autre paramètres). Sur cette base, on désire aussi savoir synthétiser des sons ayant certaines affinités structurelles, du point de vue du timbre, avec certains instruments donnés, avec lesquels le son synthétisé pourra éventuellement dialoguer en concert (c'est le cas, assez important puisque largement utilisé par les compositeurs, des oeuvres pour instruments et bande).

4.1 La synthèse additive

Il s'agit là, conceptuellement, du type de synthèse le plus simple. Avec cette technique **[MOORER, J.A., 1977]** le son est synthétisé par le contrôle, direct et indépendant, de ses composantes spectrales. Dans sa forme la plus simple, la technique est exprimée par la formule

$$(4) \qquad X(n) = \sum_{k=1}^{M} A_k(n) \sin\{n\,T\,[\omega_k + 2\pi F_k(n)] + \theta_k\}$$

où:

- $X(n)$ est le signal au temps nT (T est la période d'échantillonnage, n est un entier);
- $A_k(n)$ est l'amplitude de la k-ième harmonique;
- $\omega_k = 2\pi k F$ est la pulsation correspondante à la k-ème harmonique (F est la fréquence fondamentale du son);
- $2\pi F_k(n)$ exprime la déviation en fréquence de la k-ème harmonique;
- θ_k est l'éventuelle phase initiale de la k-ème harmonique.

$A_k(n)$ et $F_k(n)$ sont supposées être des fonctions lentes du temps (par rapport à l'évolution du signal lui-même).

La synthèse additive

L'instrument de base de cette technique est montré fig. 3.

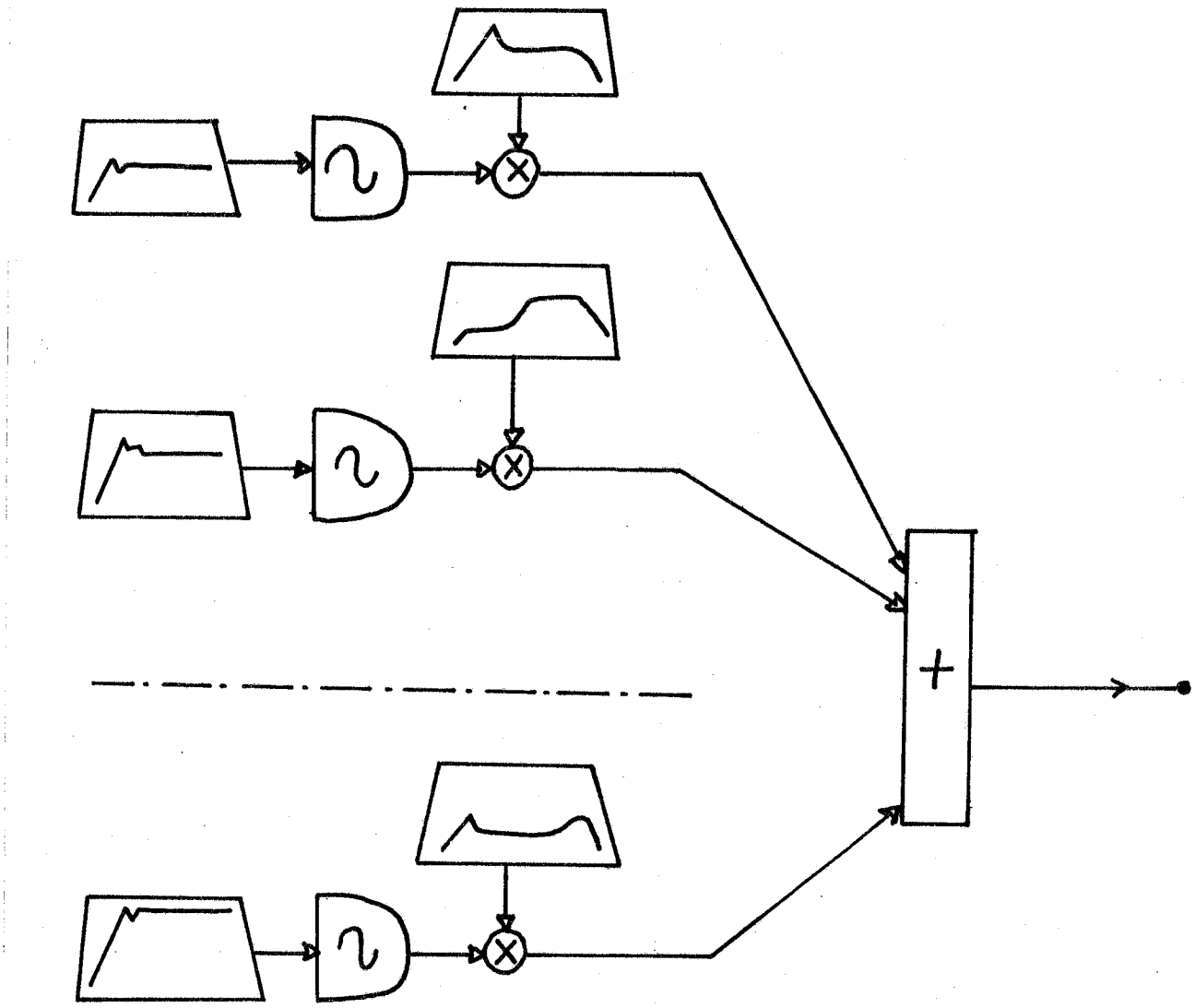

fig. 3 - instrument de base pour la synthèse additive

La formule (4), telle qu'elle est, peut servir de modèle pour une technique de synthèse des timbres naturels à partir de l'analyse. Il existe, à ce propos, plusieurs techniques pour

analyser les sons instrumentaux et en extraire les évolutions de $A_k(n)$, $F_k(n)$; cependant, nous ne détaillerons pas ce sujet [10]. Les résultats qu'on peut obtenir par cette technique sont en général très convaincants, mais le principal désavantage est que le nombre de paramètres à contrôler est souvent très grand (ce qui peut devenir un problème très sérieux si on synthétise en temps réel: l'ordinateur qui gère le synthétiseur peut ne pas être assez rapide pour calculer tous les paramètres et les sortir au fur et à mesure que le son évolue).

Ce type de synthèse, comme aussi d'autres qu'on verra, se prête à des généralisations immédiates, pour une synthèse non basée sur l'analyse: tout en laissant la formule (4) formellement inchangée, en général les fréquences composantes peuvent être choisies arbitrairement (ω_k quelconque), c'est à dire non nécessairement en relation harmonique entre elles, et aussi la forme d'onde des oscillateurs peut être différente d'une sinusoïde, en compliquant ainsi a priori le spectre résultant.

4.2 La synthèse soustractive

Selon cette technique **[CANN, R., 1979]**, le son est modulé dynamiquement à partir d'une forme d'onde spectralement complexe, envoyée en entrée à un filtre variable dans le temps; on peut donc la résumer, en général, par la formule

$$(5) \qquad X(n) = - \sum_{k=1}^{P} a_k(n)\, X(n-k) + G(n) \sum_{r=0}^{Q} b_r(n)\, U(n-r)$$

où:

- X(n) est le signal de sortie à l'instant nT;
- $a_k(n)$, $b_r(n)$ sont les valeurs des coefficients du filtre [11];
- G(n) est le gain du filtre;
- U(n) est le signal d'excitation du filtre [12].

$a_k(n)$, $b_r(n)$ et G(n) sont supposées être des fonctions lentes du temps.

L'instrument de base de cette technique est montré fig. 4.

Ce type de synthèse est très utilisé actuellement dans les studios et surtout dans les synthétiseurs analogiques traditionnels, où presque toujours (cfr. note [2]) les formes d'onde sont obtenues en envoyant un signal à onde carrée, triangulaire ou à dent de scie, obtenues par un VCO, à l'entrée d'un filtre VCF. Toutefois, dans les applications digitales, la méthode est rarement utilisée de la façon qu'on vient de rappeler, car elle est imprécise, limitée et empirique; son utilisation est aujourd'hui étroitement liée aux techniques de la synthèse numérique en temps réel

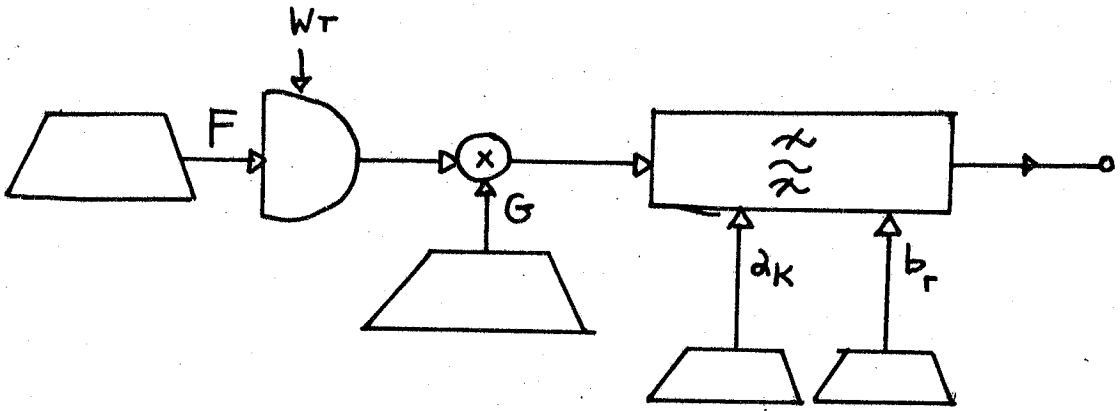

fig. 4 - instrument de base pour la synthèse soustractive. L'oscillateur et le filtre schématisés ici peuvent correspondre à l'interconnexion de plusieurs modules

synthèse de la parole **[CARRE, R., et al., 1979]**, basées sur l'analyse (surtout synthèse par formants et par prédiction linéaire **[MARKEL,J.D., GRAY, A.H., 1976; MOORER, J.A., 1978 a]**), qui permettent justement d'utiliser de nombreux algorithmes pour extraire les paramètres de l'instrument. L'objet de l'analyse, de toute façon, n'est pas limité seulement à la voix humaine: en général, cette technique se révèle très intéressante pour la synthèse de tout mécanisme de production sonore où on peut distinguer une source d'excitation connectée à un système acoustique passif, plus ou moins compliqué et normalement variable. La voix humaine fait évidemment partie de cette catégorie de processus - les cordes vocales étant la source d'excitation et le conduit vocal, variable en fonction de la position de la langue, des lèvres etc., faisant fonction de système acoustique excité [13] -, mais il en est de même pour beaucoup d'instruments traditionnels: par exemple, dans une clarinette l'anche est le système d'excitation et le corps de l'instrument fait fonction de système excité, qui peut varier ses caractéristiques en fonction de la position des clés sous les mains de l'exécutant.

Un aspect très intéressant de cette technique est que la synthèse est nettement coupée en deux parties: la production de l'excitation et le contrôle de la modulation spectrale. En

utilisant les algorithmes d'analyse dont on a parlé, on peut extraire les données pour la synthèse à partir de deux analyses différentes, concernant chacune une seule des deux parties de la synthèse en objet: on parle alors de synthèse croisée (cross synthesis), par laquelle on peut obtenir des effets particuliers comme des violons (ou des orchestres entiers) qui parlent, ou une trompette avec les modes d'attaque et d'extinction d'un piano, et ainsi de suite. De plus, il est facile, par cette technique, de contrôler la hauteur (ou pitch) et les durées du son indépendamment des autres paramètres, ce qui permet d'ultérieures et fines manipulations sur le signal synthétisé.

4.3 La synthèse par modulation de fréquence

Cette technique **[CHOWNING, J.M., 1973]**, aujourd'hui devenue très populaire, est exprimée par la formule

$$(6) \qquad X(n) = A(n) \sin[2\pi f_c\, n\, T + I(n) \sin(2\pi f_m\, n\, T)]$$

où:

- X(n) est le signal de sortie à l'instant nT;
- A(n) est l'amplitude globale du signal;
- I(n) est la valeur de l'index de modulation;
- $\omega_c = 2\pi f_c$ est la pulsation correspondante à la fréquence porteuse;
- $\omega_m = 2\pi f_m$ est la pulsation correspondante à la fréquence modulante.

A(n) et I(n) sont supposées être des fonctions lentes du temps.

L'instrument de base de cette technique est celui, déjà montré, de fig. 1, où A_m joue le rôle de index de modulation I [14].

Il est connu **[ANGOT, A., 1972]** que la formule (6) admet le développement en série

$$(7) \qquad X(n) = J_o(I) \sin\theta + \sum_{k=1}^{\infty} J_k(I)[\sin(\theta+k\beta) + (-1)^k \sin(\theta-k\beta)]$$

où, par simplicité, on a posé:

- $\theta = 2\pi f_c\, nT$;
- $\beta = 2\pi f_m\, nT$;

- I = I(n);

et où $J_k(.)$ indique la fonction de Bessel du premier type et d'ordre k.

Les formules (6) et (7) sont bien connues pour leurs applications aux techniques de transmission radio **[PERONI, B., 1973]**; dans la synthèse musicale, en dernière analyse, le procédé est exactement le même, avec pour seule substantielle différence que, et la fréquence modulante, et la fréquence porteuse sont toujours comprises dans la bande audio (alors que dans les applications radio, la porteuse est normalement une fréquence de l'ordre des centaines de MegaHertz), ce qui fait que tout le spectre ainsi formé est audible.

La formule (7) montre en pratique que le spectre d'un signal FM est constitué de plusieurs raies, aux fréquences $f_c \pm k\ f_m$, dont les amplitudes dépendent de la valeur de l'index de modulation I, par l'intermédiaire de la fonction de Bessel d'ordre k. La fig. 5 montre ces fonctions: on peut voir facilement que, pour I=0, seule la fonction d'ordre 0 est non nulle, ce qui veut dire que seule la porteuse existe dans le spectre; pour une valeur donnée de I, toutes les fonctions d'ordre plus grand que I+1 ont des valeurs suffisamment petites pour pouvoir être négligées. L'index de modulation règle donc la richesse spectrale du signal synthétisé, car plus I est grand, plus on a de raies ayant une amplitude non négligeable [15]. Les relations de phase entre ces raies sont contrôlées soit par les inversions de signe de fonctions $J_k(.)$, soit par le coefficient $(-1)^k$ présent dans la formule (7) qui intervertit la phase des raies correspondantes aux fréquences $f_c - k\ f_m$, avec k impair.

Tout cela signifie que, par les variations de deux seuls paramètres (l'index de modulation I et l'amplitude globale A), on peut engendrer des spectres aux évolutions dynamiques très complexes, avec des changements simultanés et apparemment indépendants des amplitudes de chaque composante fréquentielle: or, il a été montré **[RISSET,J.C., MATHEWS, M.V., 1969]** que l'aspect évolutif du spectre joue un rôle fondamental dans la détermination du timbre. Là réside la grande puissance et l'intérêt de la méthode, qui permet en pratique de synthétiser des timbres ayant des remarquables affinités perceptives avec les instruments traditionnels. Par contre, la méthode n'est pas très adaptée à une vraie simulation de ces instruments (par une synthèse basée sur l'analyse), car la recherche des valeurs des paramètres est un procédé essentiellement empirique.

La formule (7) peut faire croire que les spectres synthétisés par cette technique sont toujours symétriques autour de la fréquence porteuse, mais cela n'est pas vrai car, pour des valeurs suffisamment élevées de k (pour k plus grand que f_c / f_m), les fréquences des raies deviennent négatives et se reflètent donc, avec phase inversée: par exemple, si f_c = 300 Hz et f_m = 100 Hz, les raies correspondant à k=4 auront comme fréquences 300 + 4 x 100 = 700 Hz et 300 - 4 x 100 = -100 Hz;

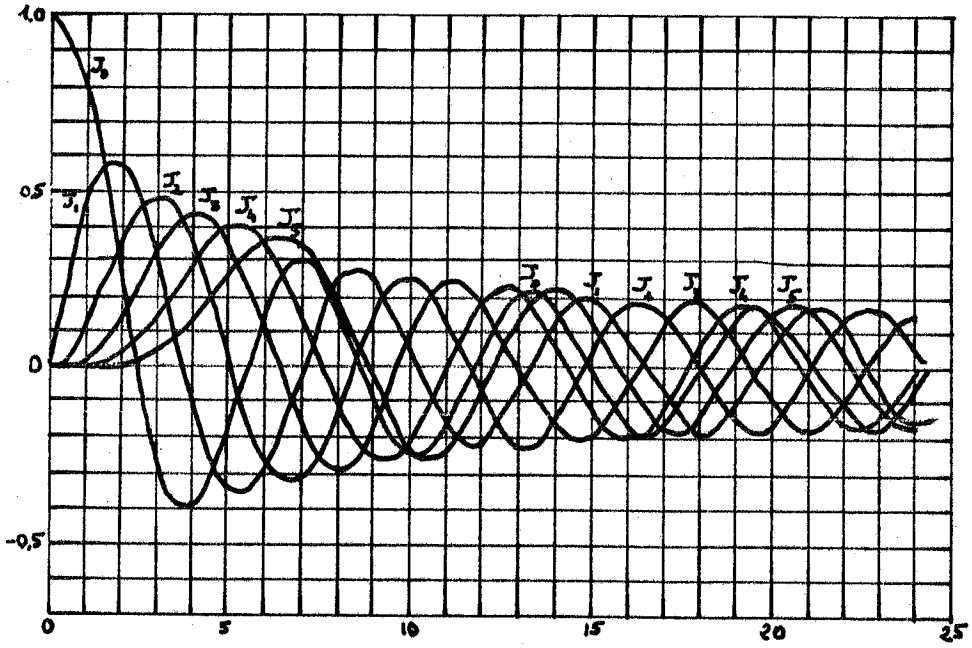

fig. 5 - fonctions de Bessel du premier type

cette dernière fréquence, toutefois, n'a qu'une existence purement mathématique et se traduit en pratique en une fréquence de +100 Hz avec phase inversée. La valeur de la composante à 100 Hz sera donc donnée, en total, par l'amplitude de la raie correspondante à k=2 (300 - 2 x 100 = 100 Hz) moins (à cause de l'inversion de phase) l'amplitude de la raie correspondante à k=4. Cela détruit donc la symétrie du spectre.

Si f_c et f_m ont un rapport du type

$$(8) \qquad f_c / f_m = c / m$$

(où c et m sont deux nombres entiers), alors

$$(9) \qquad f_c / c = f_m / m = f_o$$

et le son synthétisé sera donc quasi-périodique, avec fréquence fondamentale f_o. Toutes les raies reflétées autour de l'origine de l'axe des fréquences se superposent exactement à des autres raies déjà existantes (comme dans l'exemple qu'on vient de faire). Si par contre le rapport entre f_c et f_m est un nombre irrationnel, le spectre qui en résulte est apériodique (les fréquences reflétées tombent dans des points asymétriques entre les autres composantes), et cela peut être utilisé par exemple pour la synthèse d'instruments de type percussif.

Cette technique de synthèse peut naturellement être généralisée, en utilisant une forme d'onde quelconque pour la fréquence modulante, et (si on veut) pour la porteuse [16]; en particulier, il a été montré **[SAUNDERS, S., 1977]** que l'utilisation d'une forme d'onde triangulaire pour la modulante permet de simplifier les calculs internes à l'instrument (en éliminant une multiplication) tout en laissant les résultats pratiquement inchangés par rapport au cas de la forme d'onde sinusoïdale.

4.4 La synthèse par distorsion non linéaire

Cette technique de synthèse **[ARFIB, D., 1979; LE BRUN, M., 1979; ROADS, C. 1979]**, d'introduction assez récente, est plus générale, du moins au niveau conceptuel, que la synthèse par modulation de fréquence. Le signal est obtenu par distorsion d'un générateur sinusoïdal, qu'on fait passer par un bloc non linéaire: la formule générale est donc

$$X(n) = A(n)\ F\ [\ H(n)\ \cos\ (2\ \pi\ f\ n\ T)\ ] \qquad (10)$$

où:

- X(n) est le signal de sortie à l'instant nT;
- A(n) est l'amplitude globale du signal;
- H(n) est l'amplitude de la sinusoïde d'entrée;
- f est la fréquence de la sinusoïde d'entrée;
- F(.) est une fonction non linéaire quelconque, qu'on peut facilement réaliser par la technique de table lookup décrite à propos du bloc oscillateur: il suffit d'utiliser une mémoire où le signal d'entrée soit connecté aux lignes d'adressage.

A(n) et H(n) sont supposées être des fonctions lentes du temps; la sinusoïde d'entrée est exprimée ici par la fonction cosinus pour simplicité d'exposition.

L'instrument de base de cette technique est montré fig. 6.

Si on effectue une distorsion du signal d'entrée, on obtient donc évidemment une forme d'onde avec plusieurs harmoniques:

$$X(n) = \sum_{k=0}^{\infty} C_k \sin(\omega_k\ n\ T + \theta_k) \qquad (11)$$

où:

- C_k est l'amplitude de la k-ième harmonique;

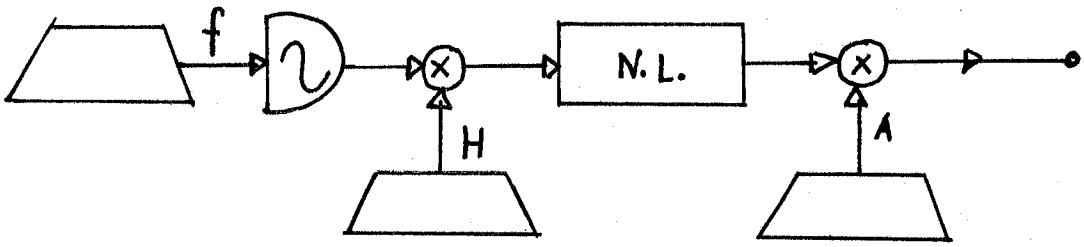

fig. 6 - instrument de base pour la synthèse par distorsion non linéaire

- $\omega_k = 2\pi k f$ est la pulsation de la k-ième harmonique;
- θ_k est la phase de la k-ième harmonique.

Les amplitudes des harmoniques dépendent naturellement, d'une façon en général très complexe, de la fonction de distorsion F(.); pour obtenir un contrôle sur ces amplitudes, on peut utiliser une classe particulière de fonctions non linéaires: la famille des polynômes de Tchebycheff du premier type, $T_k(x)$ **[ANGOT, A., 1972].**

Une des propriétés de ces polynômes est de satisfaire l'identité suivante:

$$T_k(x) = \cos (k \arccos (x)) \qquad (12)$$

Si on pose $x = \cos(\theta)$, on obtient l'intéressante propriété

$$T_k(\cos\theta) = \cos(k\theta) \qquad (13)$$

qui peut être énoncée comme suit: l'application du polynôme de Tchebycheff du premier type et d'ordre k à une fonction sinusoïdale donne comme résultat la k-ième harmonique de la sinusoïde en question.

Si donc on utilise, comme fonction non linéaire F(.), une combinaison linéaire de polynômes de Tchebycheff avec coefficients h_k

La synthèse par distorsion non linéaire

$$(14) \qquad F(.) = \sum_{k=1}^{N} h_k \, T_k(.)$$

le signal de sortie sera, chaque polynôme étant responsable d'une et d'une seule harmonique,

$$(15) \qquad X(n) = A(n) \sum_{k=1}^{N} h_k \cos(\omega_k n \, T)$$

où les coefficients h_k sont les mêmes que ceux de l'équation (14).

Dans cet exposé on a supposé, jusque là, que l'amplitude H(n) de la sinusoïde d'entrée est fixe et égale à 1; si par contre, à la limite, H(n) tend vers zéro, en supposant que la F(.) vaut 0 à l'origine des abscisses, on pourra certainement l'approximer par une ligne droite, et alors la distorsion sera nulle; si maintenant on fait varier H(n) entre ces deux cas limite, par le moyen du générateur d'enveloppe, il est évident qu'on obtiendra une variation dynamique et indépendante de chaque harmonique du signal de sortie - comme dans le cas de la modulation de fréquence -, qui variera entre une valeur d'amplitude 0 et une valeur finale, donnée par le coefficient du polynôme correspondant. En général, la formule exprimant la composition harmonique du signal de sortie est donc

$$(16) \qquad X(n) = A(n) \sum_{k=1}^{N} a_k(H) \cos(\omega_k n \, T)$$

où les $a_k(H)$ sont des fonctions qui dépendent de F(.), et qui tendent à la limite vers les valeurs h_k (si la F(.) est une combinaison linéaire de polynômes de Tchebycheff). Le problème de la détermination exacte de ces fonctions a été traité par SUEN **[C.Y., 1970]**; on peut observer que la technique de synthèse par FM peut être considérée comme un cas particulier de synthèse par distorsion non linéaire, où les fonctions $a_k(H)$ deviennent les fonctions de Bessel de l'index de modulation $J_k(I)$.

Un des inconvénients de cette technique, par rapport à la FM, est que le paramètre H, ne règle pas seulement la richesse spectrale du signal (de façon parfaitement analogue au rôle joué par l'index de modulation dans l'autre technique), mais influence aussi en partie l'amplitude resultante du signal, ce qui oblige à prendre en compte un facteur de compensation de l'amplitude dans le calcul du paramètre A.

La méthode de synthèse qu'on vient d'exposer génère intrinsèquement des spectres qui sont toujours harmoniques; plusieurs méthodes ont été proposées dans **[ARFIB, D., 1979]** et **[LE BRUN, M., 1979]** pour adapter la technique à la génération de spectres inharmoniques, méthodes qu'on ne détaillera pas ici.

5. Les synthétiseurs en temps réel

Il existe aujourd'hui, dans le monde, un certain nombre - petit - de synthétiseurs numériques fonctionnant en temps réel, contrôlés dans la plupart des cas par un mini-ordinateur, développés par des instituts de recherche en synthèse musicale.

Un des instituts de recherche qui a donné les contributions les plus intéressantes à l'évolution de ces systèmes est certainement l'IRCAM [17]; dans les pages qui suivent, on décrira avec un certain détail la structure et le fonctionnement de trois des synthétiseurs qui ont été conçus et réalisés dans cet institut, d'une part pour étudier à fond des cas particuliers et concrets, et d'autre part parce-que, il faut le dire, ces systèmes sont certainement entre les mieux réussis à l'heure actuelle, et les plus prometteurs pour l'avenir.

5.1 Le système 4A

On a déjà montré, dans la fig. 2, quels sont les circuits de base qui rentrent dans la réalisation d'un oscillateur numérique; la sortie de la mémoire de forme d'onde peut être suivie par un multiplicateur (physique), connecté à l'autre entrée à un troisième registre, dont le contenu représentera alors l'amplitude de l'oscillateur (fig. 7).

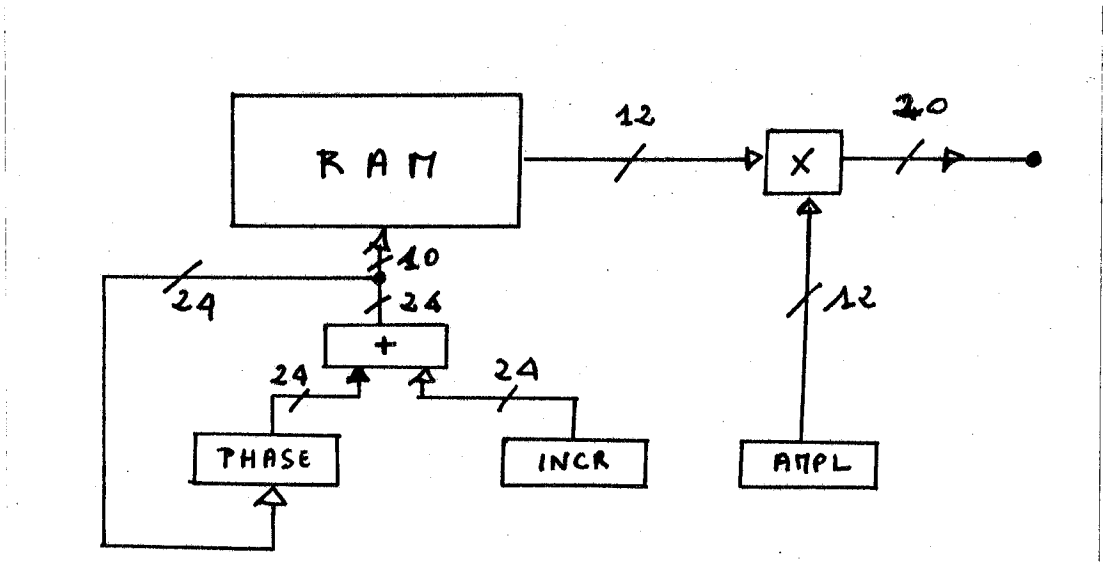

fig. 7 - oscillateur numérique avec contrôle de l'amplitude

Si maintenant on veut réaliser plusieurs oscillateurs ensemble, on peut généraliser ce schéma comme montré fig. 8: la seule différence essentielle qu'on rencontre par rapport à la figure précédente est que les trois registres sont remplacés ici par des RAMs, à N mots de 24 bits (où N est le nombre

d'oscillateurs qu'on veut réaliser); ces mémoires sont adressées par un compteur modulo N, qui est incrémenté à une fréquence opportune. Au premier cycle du système, les adresses 0 des mémoires sont sélectionnées: on additionne donc les contenus des premières "locations" (ou emplacements) de la RAM-PHASE et de la RAM-FREQ, le résultat est re-stocké dans la RAM-PHASE (toujours à la location 0) et adresse la RAM-ONDE, la valeur ainsi obtenue est multipliée par la première location de la RAM-AMPL, additionnée au contenu du registre ACCU (qui contient initialement 0) et mémorisée en ACCU; au deuxième cycle, les mêmes opérations sont effectuées à partir des contenus des locations numéro 1 des mémoires, et le résultat est accumulé en ACCU avec le résultat précédent; et ainsi de suite. Il est donc évident que le schéma de la figure permet de réaliser N oscillateurs, dont les N phases, fréquences et amplitudes sont mémorisées en N locations des trois mémoires. Après N cycles, ACCU contient le résultat de l'accumulation des N oscillateurs, qui contribuent à la valeur de l'échantillon final de sortie; à ce point, l'échantillon est mémorisé dans le registre final REG et présenté ainsi au DAC, qui en effectue la conversion en signal électrique continu; immédiatement après le stockage du résultat en REG, ACCU est remis à zéro et tout le processus recommence, pour la préparation de l'échantillon de sortie suivant. Si T est la période d'échantillonnage utilisée, il faut présenter un nouvel échantillon à REG tous les T secondes, c'est à dire qu'il faut effectuer toutes les opérations d'un cycle élémentaire en T/N secondes. Plus les circuits utilisés sont donc rapides, plus grand est le nombre d'oscillateurs qu'on peut réaliser avec une carte de synthèse du type de la fig. 8; en tout cas, le nombre de composants de la carte est fixe, et n'augmente pas avec le nombre d'oscillateurs (ce qui est une différence fondamentale par rapport aux oscillateurs analogiques).

Grâce aux circuits très performants qui existent aujourd'hui, il a été possible de réaliser une carte de ce type réalisant 256 oscillateurs en temps réel, à une fréquence d'échantillonnage de 16 kHz; cette carte est à la base du système de synthèse 4A **[DI GIUGNO, G., 1976]**. Le cycle de base est exécuté donc en

$$16000^{-1} / 256 = 62.5 \text{ usec} / 256 = 244.14 \text{ nsec}$$

Etant donné que toute la structure du synthétiseur est en temps partagé (c'est à dire, justement, que les unités de calcul effectuent le travail de plusieurs oscillateurs en divisant le temps entre un échantillon et le suivant), il est évident que 16 kHz représentent seulement une fréquence minimale d'échantillonnage, car il est toujours possible de doubler cette valeur au prix d'une réduction de la moitié du nombre d'oscillateurs: il suffit de programmer le compteur pour qu'il adresse les mémoires modulo 128, recommençant donc le cycle complet deux fois plus vite. Ce procédé peut être itéré, donnant lieu à des fréquences d'échantillonnage de plus en plus élevées.

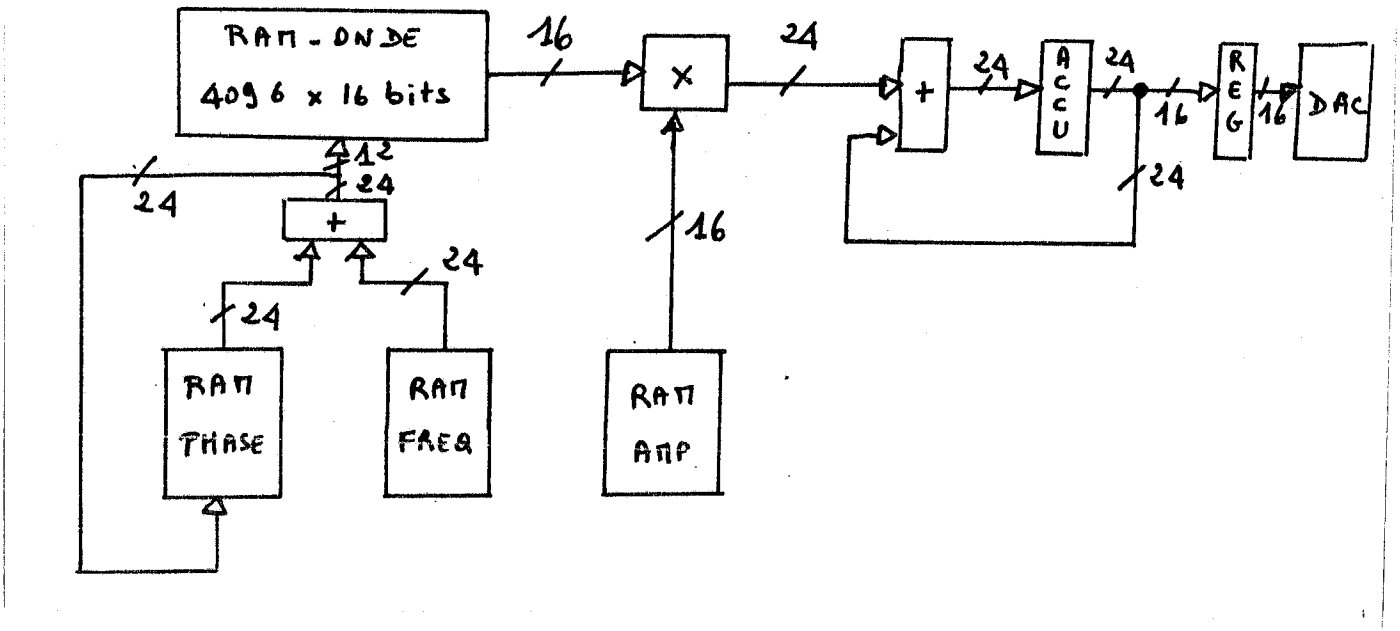

fig. 8 - schéma physique de la 4A

Des valeurs de 64 kHz ou plus peuvent sembler excessives pour des applications musicales, mais disposer de ces valeurs s'est montré très utile dans certains cas où les oscillateurs avaient des formes d'onde très riches d'harmoniques et où il fallait soigneusement éviter toutes distorsions dues au repliement du spectre [18].

Les mémoires RAM-ONDE, RAM-FREQ, RAM-AMPL et éventuellement RAM-PHASE (si on veut un contrôle sur la phase initiale) sont accessibles par l'ordinateur pilote, qui peut ainsi programmer au préalable la forme d'onde (qui est la même pour tous les oscillateurs) et contrôler ensuite, en temps réel, les fréquences et les amplitudes de chacun des oscillateurs. On remarquera qu'il n'y a pas de générateurs d'enveloppe prévus dans la carte, donc toutes les évolutions des paramètres doivent être fournies, à une cadence adéquate, par l'ordinateur.

En bref, le synthétiseur 4A se présente comme un bloc de 256 oscillateurs, avec contrôle de fréquence et d'amplitude, accumulables entre eux. La fig. 9 montre le schéma fonctionnel du système, qui est évidemment destiné surtout à la synthèse par la technique additive.

Le système 4A

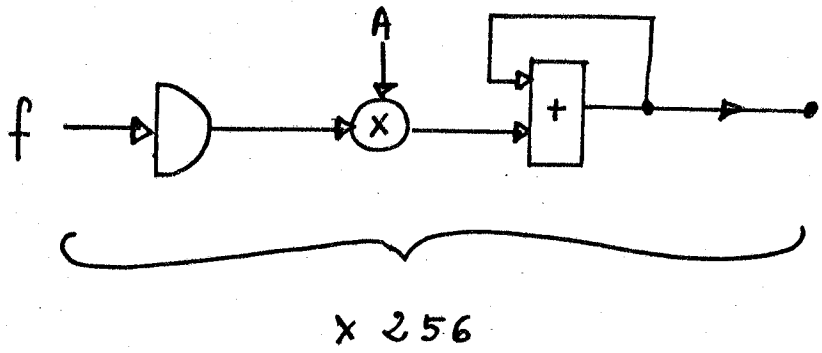

fig. 9 - schéma fonctionnel de la 4A

On peut remarquer facilement que, dans ce système de synthèse comme dans les autres qu'on décrira, le problème de l'utilisation des ressources se pose de façon très différente que pour les systèmes en temps différé: dans ce dernier cas, l'importance des calculs pour la synthèse du signal est en rapport avec le nombre d'unités utilisées dans la définition de l'instrument; il n'existe pas de limitation au degré de complexité qu'on peut atteindre, mais plus l'instrument est complexe, plus la phase de calcul du signal est lente; dans le cas du temps réel, par contre, utiliser moins d'oscillateurs n'accélère en rien le processus de calcul, puisque le synthétiseur est aveugle et que les opérations sont faites de toute façon; on peut donc aller sans problèmes jusqu'à utiliser tous les oscillateurs, et le résultat est toujours sorti en temps réel; en revanche, il n'est absolument pas possible d'aller au-delà de ce nombre maximum.

L'ordinateur qui pilote le système est un mini-ordinateur PDP-11/55 de Digital Equipment Corporation; le synthétiseur a été conçu de toute façon pour être interfacé sur l'Unibus, qui est le bus standard de tous les PDP-11 [19]: n'importe quel ordinateur de cette famille peut donc être utilisé pour gérer le système. Cet ordinateur a été choisi en raison de sa flexibilité et de sa facilité d'interfaçage. Le synthétiseur est vu par l'ordinateur comme une partie de sa mémoire centrale, ce qui évite de perdre du temps à échanger des données entre la mémoire interne du PDP-11 et les mémoires du synthétiseur; toutes ces mémoires peuvent être ainsi manipulées par n'importe quelle instruction de l'ordinateur.

5.2 Le système 4C

L'architecture de la 4C **[MOORER, J.A., et al., 1979; CAPPIELLO, C., 1980]** est nettement plus complexe que celle de la

4A, ce qui entraîne une souplesse et une puissance d'utilisation d'autant plus poussées; les unités de base sont à un niveau plus bas (même un bloc aussi commun que l'oscillateur n'est pas déjà prêt, comme dans la 4A, mais il doit être programmé par l'utilisateur), et en ce sens on peut dire qu'on remonte un peu vers l'ordinateur intelligent, moins spécialisé.

D'un point de vue structurel, la 4C se présente comme un ensemble d'éléments arithmétiques/logiques, de mémoires et de registres qui sont contrôlés par un microprogramme interne, qui réalise certaines unités élémentaires à partir desquelles, par programmation du système, tous les blocs de synthèse doivent être construits.

Ces unités de base, illustrées fig. 10, sont les suivantes:

- 64 additionneurs à trois entrées avec lecture de tableau; ils effectuent la somme de trois nombres, et en même temps une des entrées est aussi utilisée pour adresser une mémoire, qui contient normalement les formes d'onde des oscillateurs;

- 64 multiplicateurs-additionneurs; ils effectuent le produit entre deux nombres et ajoutent un troisième nombre au résultat;

- 32 unités logiques; elles effectuent l'addition entre deux nombres et comparent la somme avec un troisième; le résultat final peut être la somme ou bien le dernier nombre, en fonction de la comparaison; en outre, à chaque unité logique est associé un ultérieur multiplicateur-additionneur;

- 32 timers; ils consistent en un registre qui se décremente d'une unité à intervalles de temps fixes (et programmables); quand un timer arrive à zéro, il peut éventuellement interrompre le déroulement normal du programe de l'ordinateur maître et l'obliger à exécuter une routine de service particulière.

Le nombre des unités disponibles dans le système est en rapport avec la fréquence d'échantillonnage de 16 kHz, étant entendu que, comme dans la 4A, on peut toujours doubler cette fréquence au prix d'une réduction à la moitié des unités.

Toutes les entrées et les sorties de chaque unité sont interconnectables entre elles de façon complètement arbitraire et décidable par l'utilisateur; on peut voir que, moyennant des interconnexions opportunes de ces unités, on peut réaliser pratiquement tous les blocs de synthèse qui rentrent dans les techniques décrites dans les chapitres précédents: oscillateurs, filtres, instruments à modulation de fréquence, fonctions non linéaires, et encore générateurs de bruit, modulateurs d'amplitude, etc.; l'utilisation typique des unités logiques est la réalisation, avec un timer, de générateurs d'enveloppe à

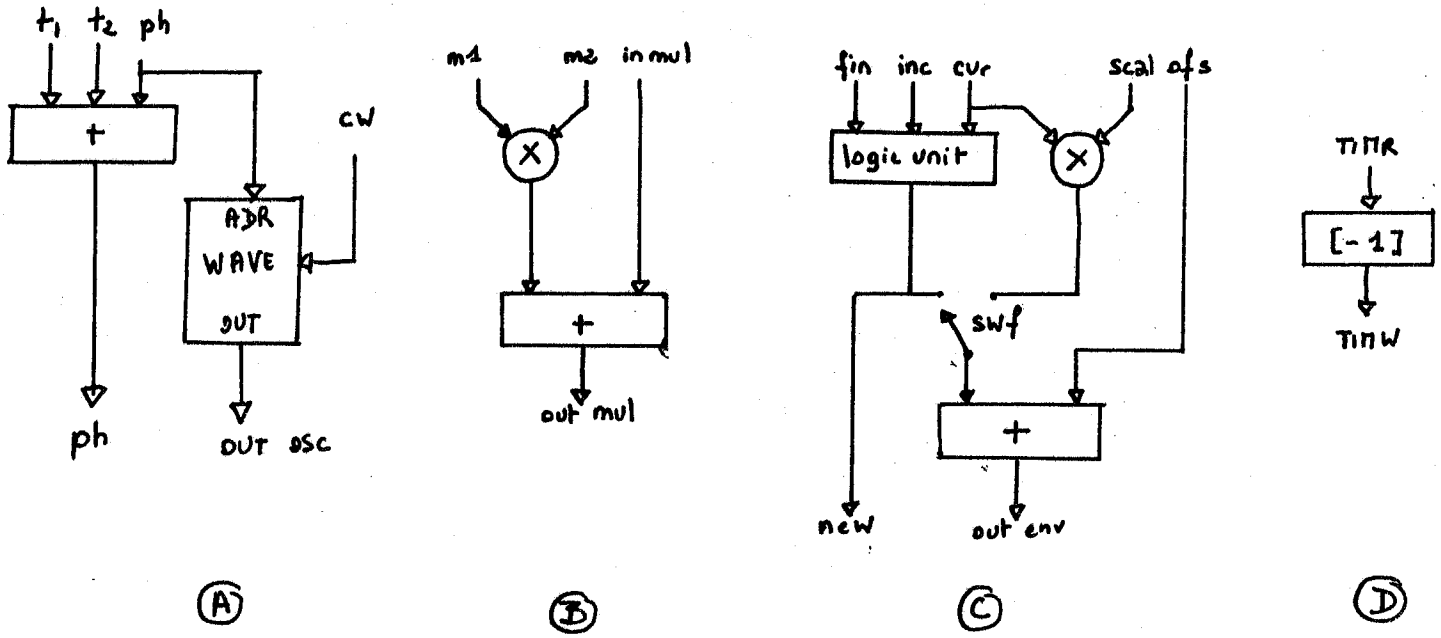

fig.10 - unités de base de la 4C. Le bloc "logic unit" exécute l'opération suivante: new = min[(cur+inc),fin] (inc $\geq$ 0); new = max[(cur+inc),fin] (inc < 0) (voir Appendice II).

segments, mais on peut aussi s'en servir pour la préparation de différents blocs particuliers comme des redresseurs, des intégrateurs avec remise à zéro, des détecteurs de mélodie (pitch detectors), des sélecteurs de maximum, et ainsi de suite; des détails sur la réalisation de ces modules sont donnés dans les Appendices II et III.

Il est clair que cette architecture de base rend le système d'une utilisation plus complexe [20], puisque toute utilisation, même simple et banale, exige une programmation préalable, mais on peut voir par contre que les avantages obtenus sont assez importants: non seulement on peut faire beaucoup plus de choses qu'avec une 4A, mais surtout on peut optimiser, à la limite dynamiquement - pendant une performance du système -, les ressources dont on dispose: si en effet les mêmes unités élémentaires permettent de réaliser un oscillateur, ou un filtre ou un bloc de distorsion non-linéaire, on pourra alors programmer le système pour avoir plus d'unités d'un certain type et moins d'autres, ou vice-versa, en fonction de la technique de synthèse utilisée à un moment donné.

En Appendice II on trouvera une description technique plus détaillée de la 4C.

5.3 Le système 4X

Les deux systèmes 4A et 4C consistent chacun, physiquement, dans une carte mesurant 25 x 30 cm. environ et contenant jusqu'à 162 circuits intégrés numériques; tous les efforts de conception ont été concentrés, dans le projet de ces machines, sur l'optimisation des ressources qu'on peut implanter dans une carte du type décrit.

Le système 4X **[ASTA, V., et al., 1980; DI GIUGNO, G., KOTT, J., 1980a, 1980b et 1981]**, par contre, va au-delà de cette optique, étant constitué de 11 cartes, communiquant entre elles, chacune avec une puissance de calcul comparable à celle d'une 4C. Il s'agit donc d'un système beaucoup plus gros et naturellement plus performant que les deux autres que l'on vient de voir. La fig. 11 montre l'architecture générale de la 4X [21].

Plus précisément, les 11 cartes composant le système se divisent en:

- 8 cartes de synthèse, toutes identiques, représentant en pratique une évolution du synthétiseur 4C et pouvant communiquer entre elles par un bus interne (INT-BUS);
- une carte d'interface, réalisant l'interconnexion avec l'ordinateur qui gère le système;
- 2 cartes de contrôle (réduites à une seule dans la version industrielle; voir note [21]), destinées en général à gérer les cartes de synthèse.

Chaque carte de synthèse, appelée 4U, est une généralisation de la 4C, dans le sens qu'elle a, au niveau de la structure hardware (c'est-à-dire de la structure des circuits physiques), des possibilités additionnelles (comme la possibilité d'écrire, et non seulement de lire, dans la mémoire des formes d'onde), mais surtout parce-qu'elle peut être microprogrammée par l'utilisateur: elle peut fonctionner comme une 4C, mais si on change le microprogramme on peut obtenir une carte avec d'autres unités de base, qu'on pourra choisir en vue de la mise en oeuvre d'un certain type de blocs de synthèse. Par exemple, on peut microprogrammer une 4U pour réaliser une carte comme la 4A, ou encore on peut "spécialiser" la carte pour réaliser exclusivement des filtres numériques, ou autre. Le microprogramme n'est plus figé, comme dans la 4C, mais il réside dans une mémoire qui est accessible par l'ordinateur maître et qui donc peut être changée à tout moment (voir dans l'Appendice III des exemples de microprogrammes possibles et les unités élémentaires de base qu'on peut en obtenir). En outre, une caractéristique très importante des 4U, par rapport à la 4C, est qu'elles peuvent échanger des données entre elles, tout au long du processus de synthèse: par exemple, une carte 4U microprogrammée pour réaliser une 4A peut être utilisée pour implanter un banc d'oscillateurs accumulés entre eux; la sortie de cette carte peut être envoyée

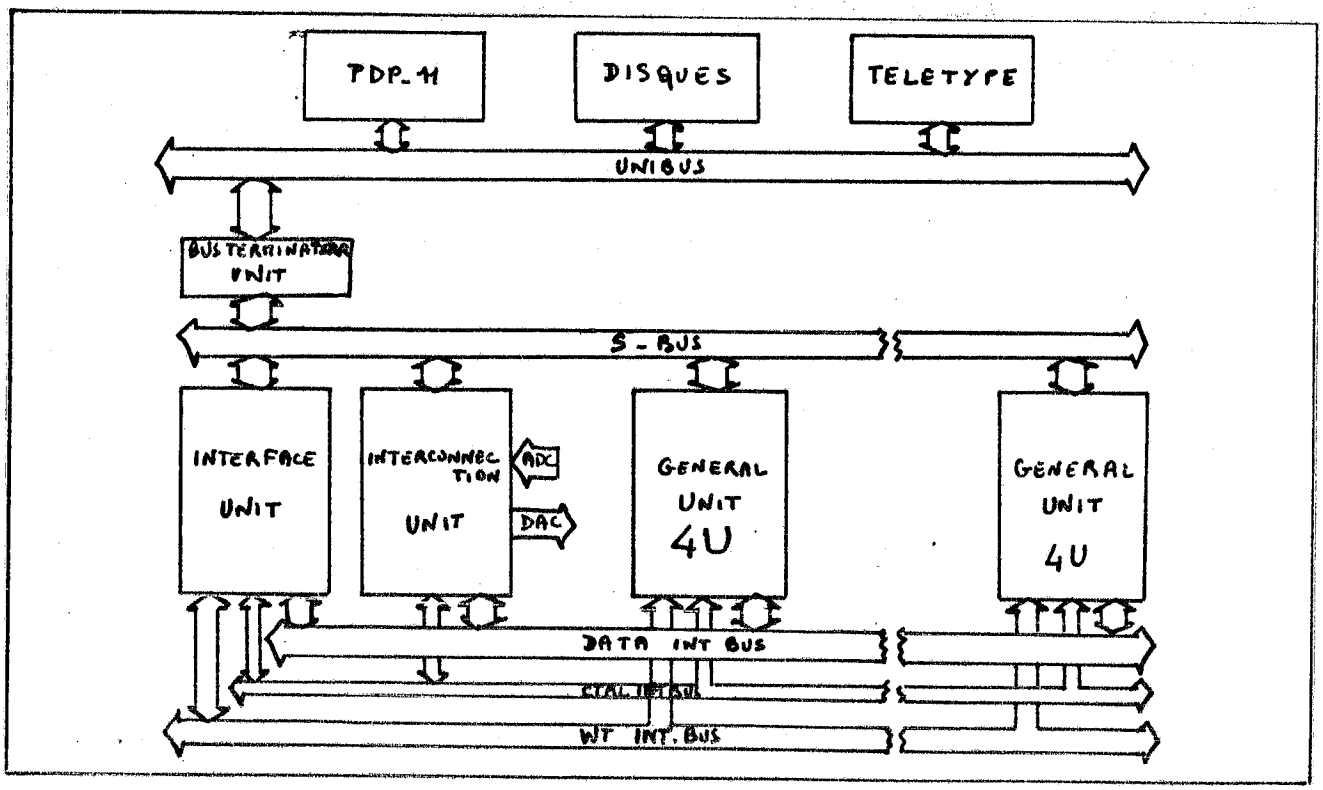

fig.11 - architecture de la 4X et structure des bus

systématiquement, au taux d'échantillonnage prévu, à une deuxième 4U destinée à la réalisation de filtres, qui effectueront sur elle des manipulations de type soustractif. La sortie de l'ensemble des filtres sera enfin envoyée aux convertisseurs. Il est facile de se convaincre que les possibilités d'applications

qui en résultent sont incomparablement supérieures à celles des modèles précédents.

La carte d'interface (Bus Terminator Unit) contient simplement l'interface hardware avec l'ordinateur maître; elle peut éventuellement se présenter sous différentes formes, si plusieurs ordinateurs sont envisagés pour le contrôle du système (dans le prototype, l'interface dont on dispose est celui pour le PDP-11/55).

Des deux cartes de contrôle, la première (Interface Unit) est chargée de gérer l'interface avec l'ordinateur au niveau de dialogue logiciel; elle contient aussi la logique pour la sélection des mémoires des 4U dans l'espace de travail du PDP-11, et une mémoire-tampon spéciale qui permet des échanges de données très rapides entre la 4X et l'ordinateur. La dernière carte (Interconnection Unit), enfin, gère le bus interne du système (signaux de contrôle du INT-BUS: voir fig. 11) en décidant de quelle carte à quelle autre faut-il transférer les données. Les cartes susceptibles d'être intéressées à ces transferts sont les 8 4U, comme on l'a déjà dit, et l'Interconnection Unit elle-même, pour ce qui concerne les convertisseurs d'entrée (ADCs) et de sortie (DACs), qui sont contrôlés par cette carte; il peut y avoir jusqu'à 16 DACs et 16 ADCs dans le prototype, ou jusqu'à 64 ADCs et DACs au total dans la version industrielle. Enfin, cette carte contient aussi tous les timers pour les 4U.

En Appendice III on trouvera une description technique plus détaillée de la 4X.

6. Les produits commerciaux

Nous ne pouvons pas terminer cet exposé sans mentionner la miriade de produits commerciaux, issus du dévelopement des techniques numériques, qui peuplent depuis quelques années le marché de la musique électronique pop.

Il s'agit, bien entendu, de systèmes beaucoup plus limités (en termes de nombre d'unités élémentaires de synthèse et de flexibilité de configuration) que ceux que l'on a décrit dans les pages précedentes, étant donné les sévères contraintes de prix auxquelles ils sont soumis, pour pouvoir viser un nombre le plus vaste possible d'acheteurs potentiels. A titre d'exemple, le système DX-7 (voir plus loin) possède 96 oscillateurs et générateurs d'enveloppe, mais on ne peut avoir qu'un maximum de 16 notes en même temps, toutes nécessairement avec le même timbre: donc, en réalité, on dispose en pratique de 6 oscillateurs et générateurs d'enveloppe, programmables, qui peuvent générer jusqu'à 16 notes différentes. Et il s'agit déjà d'un des systèmes de plus haut niveau.

La liste de produits qui suit est, naturellement, non-exhaustive; la situation du marché demeure tellement vierge et

peu explorée, que l'introduction de nouveaux produits est presque matière de tous les jours.

1. synthétiseurs

   i. synthétiseurs purement digitaux: il en existe désormais plusieurs types, mais les plus intéressants sont très probablement les modèles de la famille DX, introduite par Yamaha: DX-1, DX-7, DX-9. Fondamentalement, chacun de ces trois synthétiseurs permet d'utiliser la technique de synthèse par modulation de fréquence, avec des variantes plus ou moins sophistiquées, et avec un bon contrôle gestuel (à partir d'un clavier classique) des paramètres de synthèse.

   ii. synthétiseurs hybrides: ces systèmes comprennent généralement des oscillateurs réalisés digitalement, connectés à des amplificateurs et filtres contrôlés en tension (VCA, VCF).

   iii. synthétiseurs à usage spécial (avec accès limité)

      a. batteries électroniques: les principaux modèles existant sur le marché se basent sur des sons enregistrés et stockés dans des mémoires mortes (ROM). Ils ont tous, généralement, des difficultés évidentes pour reproduire le son des cymbales, qui sont beaucoup trop longs par rapport aux autres et qui demanderaient donc une quantité de mémoire très grande.

      b. synthétiseurs à preset: les systèmes de cette catégorie ne permettent pas de manipuler les paramètres de synthèse du son, qui sont tous stockés une fois pour toutes par le fabriquant. On a donc le choix entre un nombre limité de sons figés, ce qui rend ces instruments beaucoup plus semblables à des orgues électroniques qu'à des véritables synthétiseurs. D'autre part, leur prix est sensiblement inférieur à ceux des synthétiseurs programmables.

2. Instrumentation de support pour les synthétiseurs (analogiques ou digitaux)

   i. séquenceurs: il s'agit de petits systèmes permettant de mémoriser les mélodies jouées sur des synthétiseurs, et de les restituer ensuite; les modèles les plus sophistiqués fournissent aussi quelques possibilité d'édition (modification, copie etc.) des séquences mémorisées. Les séquenceurs analogiques, comme on l'a déjà dit (voir paragraphe "Musique et ordinateurs"), disposent de mémoires (analogiques) ayant quelques dizaines de pas au maximum, mais les techniques

digitales ont permis de construire des modèles, avec des mémoires numériques, ayant une capacité de plusieurs milliers de notes.

3. instruments pour studios d'enregistrement

   i. magnétophones numériques: Il s'agit là d'une catégorie de produits qui vient tout juste de faire ses premiers pas dans le marché, mais qui semble être très prometteuse pour l'avenir. Il est probable qu'on assistera, d'ici peu, à une véritable révolution des méthodes de travail dans les studios d'enregistrement.

4. instruments pour le traitement à posteriori du signal (synthétisé ou non)

   i. écho digital: la réalisation d'un écho par des techniques numériques est assez simple; les modèles présents sur le marché offrent, en général, 1 seconde d'écho à 25 kHz de fréquence d'échantillonnage.

   ii. réverbération artificielle: l'effet de réverbération est obtenu par des bancs de filtres numériques particuliers (voir Appendice III, paragraphe "Microprogrammes et applications")

   iii. équaliseurs numériques: Les équaliseurs analogiques classiques sont dits "paramétriques" car ils disposent de 3 ou 4 bandes (filtres passe-bande en parallèle), chacune avec 3 paramètres variables: la fréquence centrale de la bande passante, le gain, et la pente en bande de transition; naturellement, un banc de filtres numériques peut réaliser un système extrêmement plus flexible et sophistiqué.

   iv. vocodeurs: ces instruments, passés depuis longtemps déjà du domaine de la téléphonie expérimentale à celui de la musique, sont des petits systèmes d'analyse-modification-resynthèse du son, permettant éventuellement de réaliser des synthèses croisées (voir paragraphe "La synthèse soustractive").

   v. échantillonneurs

      a. échantillonneurs à mémoire: ils consistent tout simplement en un système d'acquisition d'un signal en temps réel, mémorisé et qu'on peut en suite restituer à des différentes fréquences d'échantillonnage, pour en varier la fréquence fondamentale (mais aussi la durée). Ces produits sont parfois intégrés dans les synthétiseurs numériques.

b. échantillonneurs instantanés: ces systèmes, mieux connus sous le nom de "harmonizer", effectuent le même travail (changement de fréquence fondamentale) que les échantillonneurs à mémoire, mais ils le font en temps réel (et donc sans altération de la durée).

Tous ces instruments ont acquis, récemment, un intérêt supplémentaire du fait de l'introduction de l'interface MIDI **[IMUG, 1983]** (Musical Instrument Digital Interface), c'est à dire d'un interface standard permettant de relier plusieurs instruments (synthétiseurs, séquenceurs, batteries électroniques etc.) de différents fabriquants entre eux et éventuellement à un ordinateur personnel. Cet interface a été défini par l'IMUG (International MIDI User's Group), qui est une association regroupant les plus grands fabriquants mondiaux, et, bien qu'il ait des sérieuses limitations, qui en excluent l'usage en dehors d'un contexte de musique pop, il représente néanmoins un premier pas important vers l'interchangeabilité et la compatibilité des produits d'un marché en pleine expansion.

Remerciement

Je désire remercier avant tout Peppino Di Giugno, le "papa" de ces machines prodigieuses, et Luciano Berio, qui en a appuyé la réalisation à l'IRCAM.

Merci, enfin, à Carmelo Cappiello, pour la collaboration offerte dans la préparation de ce texte, et (dulcis in fundo) à Michèle Castellengo, qui a accepté de réviser le manuscrit et qui n'a jamais arrêté de m'incourager durant tout le travail.

APPENDICE I : Listing d'un programme-exemple réalisant un oscillateur

```
C subroutine-exemple, en langage FORTRAN, pour montrer le
C fonctionnement d'un oscillateur logiciel.
C la subroutine fournit, à chaque appel, la valeur du prochain
C échantillon d'un oscillateur.
C explication des arguments de la subroutine:
C IOUT   = valeur de l'échantillon, en sortie
C H      = valeur de l'incrément pour la fréquence voulue. l'incrément
C             est calculé selon la formule (2') par le programme qui
C             appelle et il est mis à jour en fonction de la partition
C PHASE  = phase instantanée de l'oscillateur. elle est initialisée
C             par le programme qui appelle à la valeur de la phase
C             initiale de l'oscillateur, après quoi elle est incrémentée
C             systématiquement dans cette routine
C IWAVE  = tableau en mémoire contenant la forme d'onde assignée à
C             l'oscillateur. la forme d'onde a été calculée préalablement
C             par le programme qui appelle
C LENWAV = longueur du tableau de la forme d'onde

          SUBROUTINE OSCIL(IOUT,H,PHASE,IWAVE,LENWAV)
          DIMENSION IWAVE(LENWAV)

C incrément de la phase (en virgule flottante)
          PHASE=PHASE+H
C contrôle pour adressage modulo LENWAV
          IF(PHASE .GE. FLOAT(LENWAV+1)) PHASE=PHASE-FLOAT(LENWAV)
C conversion de la phase en adresse du tableau, par tronquage
C (voir note [0] pour modification pour arrondi et interpolation)
          IADR=IFIX(PHASE)
C adressage du tableau et détermination de l'échantillon de sortie
          IOUT=IWAVE(IADR)
          RETURN
          END
```

APPENDICE II : Description technique de la 4C

La 4C est un processeur très rapide spécialisé pour la synthèse et le traitement du signal acoustique en temps réel. Physiquement, elle est constituée d'une carte de 25 x 30 cm. environ contenant 162 circuits intégrés numériques, pour la pluspart en technologie TTL Shottky.

Un schéma très réduit de l'architecture interne de la 4C est montré fig. 12; le noyau central est constitué par deux mémoires (mémoire des données ou Data Memory et mémoire des adresses ou Address Memory), et par une unité de calcul (Unité Fonctionnelle ou Functional Unit).

La Functional Unit est constituée par un ensemble d'éléments arithmétiques ou logiques, plus ou moins câblés entre eux. Extérieurement elle se présente comme un bloc capable d'exécuter un certain nombre d'opérations de différents types (additions, multiplications, décisions logiques), qui seront effectuées dans une succession établie en fonction d'un microprogramme, écrit dans la mémoire de programme ou Program Memory (qui est une ROM à mots de 40 bits).

L'unité est complétée en outre par une mémoire des formes d'onde ou Waveform Memory, destinée à stocker les formes d'onde des oscillateurs, et qui de toute façon peut être vue comme une partie de la Functional Unit.

Les opérations du système sont contrôlées par le micro-programme grâce à une série de bus internes et de registres de stockage temporaire, en une succession qui se répète cycliquement tous les 62.5 usec. Dans cet intervalle de temps la Functional Unit effectue 1024 opérations de lecture des opérandes ou d'écriture des résultats dans la Data Memory. La Program Memory est constituée de 32 locations de mémoire, chacune desquelles contient - codée - une micro-instruction pour la Functional Unit.

Les opérations qui sont effectuées dépendent du microprogramme contenu dans la Program Memory, qui, en contrôlant les signaux de commande des bus et des registres, fournit les instructions à la Functional Unit. La succession des opérations est contrôlée par une horloge pilote (Master Clock) fonctionnant à 16.384 MHz.

Les opérations de lecture/écriture se font par l'intermédiaire de l'Address Memory, qui sélectionne le registre de la Data Memory dans lequel aller lire ou écrire la donnée. L'Address Memory est constituée de 1024 octets, et est adressée cycliquement par un compteur, en synchronisme avec la Program Memory. Elle peut adresser jusqu'à $2^8 = 256$ registres de données différentes (telle en effet est la dimension de la Data Memory). A chaque fois qu'on sélectionne une adresse dans la Program Memory, on fait de même pour l'Address Memory; puisque la première de ces deux mémoires établit le type d'opération à

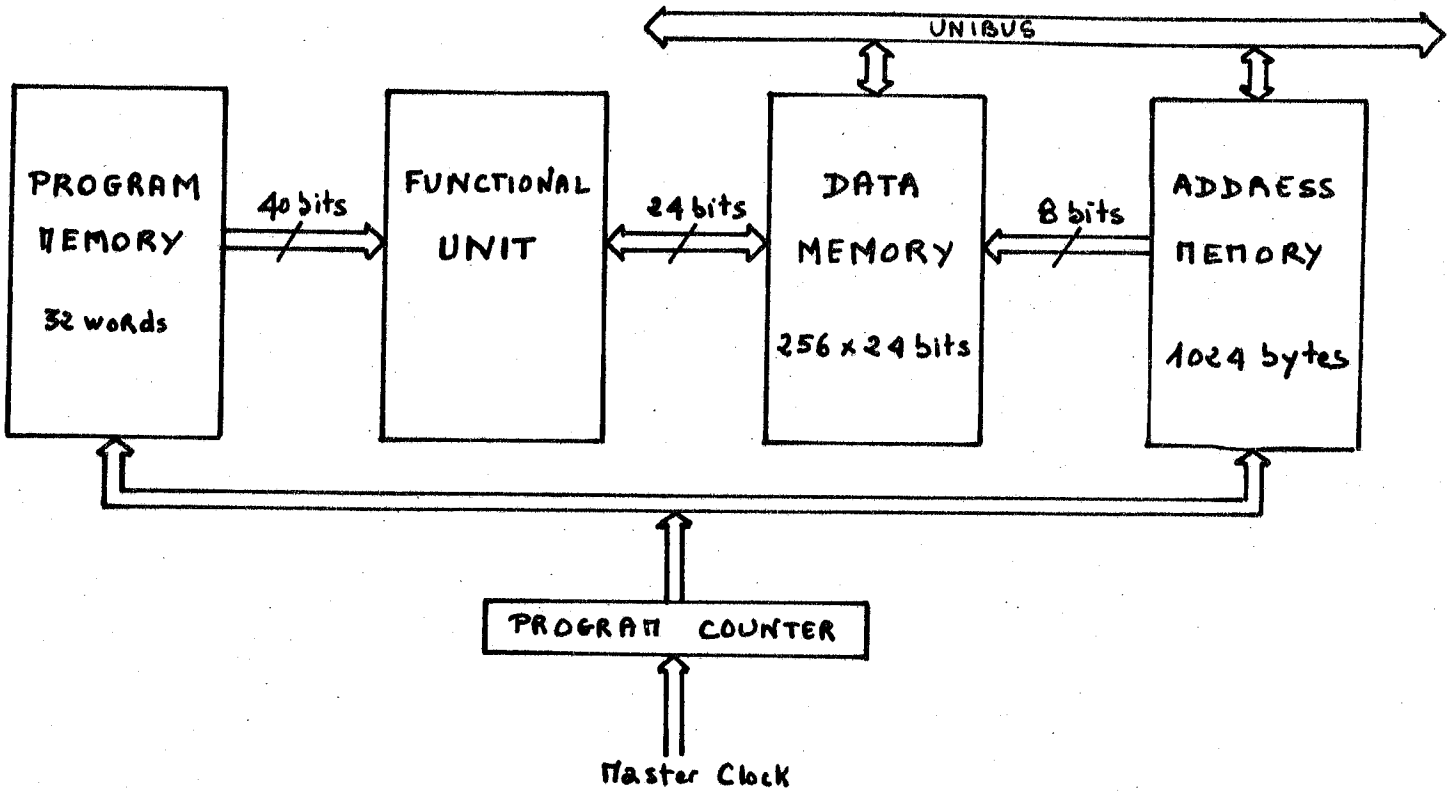

fig. 12 - les mémoires et la Functional Unit de la 4C

effectuer, et la deuxième la donnée à utiliser, avec cette méthode toute location de l'Address Memory identifie une opérande donnée d'une certaine opération.

Supposons par exemple que le système soit microprogrammé de façon à accepter deux données dans les microcycles 0 et 1 pour effectuer la multiplication entre elles, et pour rendre le résultat en sortie pendant le cycle 6. Si on écrit dans l'Address Memory les numéros 50, 55 et 110 aux adresses 0, 1 et 6, l'unité en question multipliera les données contenues dans les registres d'adresses 50 et 55 de la Data Memory et écrira le résultat dans le registre d'adresse 110 de la même mémoire. Ce résultat sera ensuite disponible comme entrée d'un élément de calcul, commençant après le microcycle n. 6. Pour l'utiliser il suffira d'écrire encore l'adresse 110 dans la location de l'Address Memory correspondant à l'un des cycles d'entrée des donnés de l'élément en question.

Comme on peut le remarquer, ce sont les adresses des opérandes, et non les opérandes elles-mêmes, qui sont associées aux opérations élémentaires de la Functional Unit; l'avantage de ce procédé est que de cette façon une certaine donnée peut être en même temps une opérande pour plusieurs opérations, et le résultat d'une opération peut être sélectionné comme opérande pour une opération ultérieure: pour ce faire, en général, il suffit de mettre son adresse dans les locations correspondantes de l'Address Memory, et la donnée sera sélectionnée, à des instants de temps différents, pour des opérations différentes. C'est avec cette technique qu'on réalise, par l'intermédiaire de la Data Memory, toutes les interconnexions internes entre les unités élémentaires de la 4C.

La structure de la Functional Unit est conçue de façon à permettre d'effectuer plusieurs opérations en même temps, si bien qu'elle peut accepter des opérandes pour une nouvelle opération avant de sortir tous les résultats des opérations précedentes (structure en pipe-line).

Etant donné que les dimensions des mémoires de programme et des adresses sont respectivement 32 micro-instructions et 1024 octets, il est évident que la première sera lue 32 fois dans le temps nécessaire pour lire entièrement la seconde. Pour eviter des confusions on appellera:

- microcycle de base l'intervalle de temps entre deux impulsions successives du Master Clock, soit 61.035 nsec;

- cycle intermédiaire le temps d'adressage de toute la Program Memory, donc l'ensemble de 32 microcycles de base, soit 1.95 usec;

- cycle complet le temps d'adressage de toute l'Address Memory, donc l'ensemble de 32 cycles intermédiaires, ou de 1024 cycles de base, soit 62.5 usec.

Il est clair que dans un cycle complet on aura 32 fois la même séquence d'opérations, mais grâce aux dimensions de l'Address Memory les opérandes, en général, ne seront pas les

mêmes.

La multiplication que nous venons de décrire sera exécutée une fois par cycle complet. Ainsi toutes les 62.5 usec (la durée de 1024 microcycles) se déroulent plusieurs opérations, multiplexées dans le temps, de traitement digital ou de synthèse du signal, avec un taux d'échantillonage de 16 kHz. Ce qui est fait dépend exactement du contenu de la Program Memory qui spécifie les opérations à effectuer, et des valeurs écrites initialement dans l'Address Memory, laquelle réalise, comme nous venons de le voir, les interconnexions entre les différents éléments de calcul. En général, programmer l'unité signifie écrire des valeurs convenables dans l'Address Memory, compte tenu du microprogramme exécuté et des interconnexions qu'on désire réaliser.

La fig. 13 montre plus en détail la structure physique interne de la 4C. On peut remarquer que les éléments qui composent la Functional Unit sont essentiellement:

- une unité logique-arithmétique (ALU), utilisée pour réaliser des additionneurs et des unités logiques;

- un multiplicateur rapide;

- la Waveform Memory, qui consiste en 16 k mots de 16 bits qu'on peut adresser soit comme un bloc unique, soit fractionnée en blocs mineurs.

  On peut avoir:

  1 bloc de 16 k mots, ou
  2 blocs de 8 k mots, ou
  4 blocs de 4 k mots, ou
  8 blocs de 2 k mots, ou
  16 blocs de 1 k mots

  Il est possible aussi d'opérer des fractionnements mixtes.

Les unités élémentaires qu'on obtient ainsi, avec le microprogramme adopté, sont les suivantes (voir fig. 10):

- additionneur à trois entrées avec lecture de tableau; il accepte en entrée trois opérandes à 24 bits et restitue, en première sortie, la somme modulo 2^{24} de ces trois nombres; en même temps, une des trois entrées est utilisée pour adresser la Waveform Memory, dont l'échantillon sélectionné est produit en deuxième sortie (sur 16 bits). Un mot de contrôle opportun, propre à chaque unité, précise quel bloc de la Waveform Memory est à considérer pour l'adressage. Le microprogramme réalise 2 de ces unités en un cycle intermédiaire, soit 64 dans un cycle complet.

- multiplicateur-accumulateur: il accepte en entrée deux opérandes à 16 bits (les plus significatifs des 24 du bus interne à la sortie de la Data Memory) et en calcule le

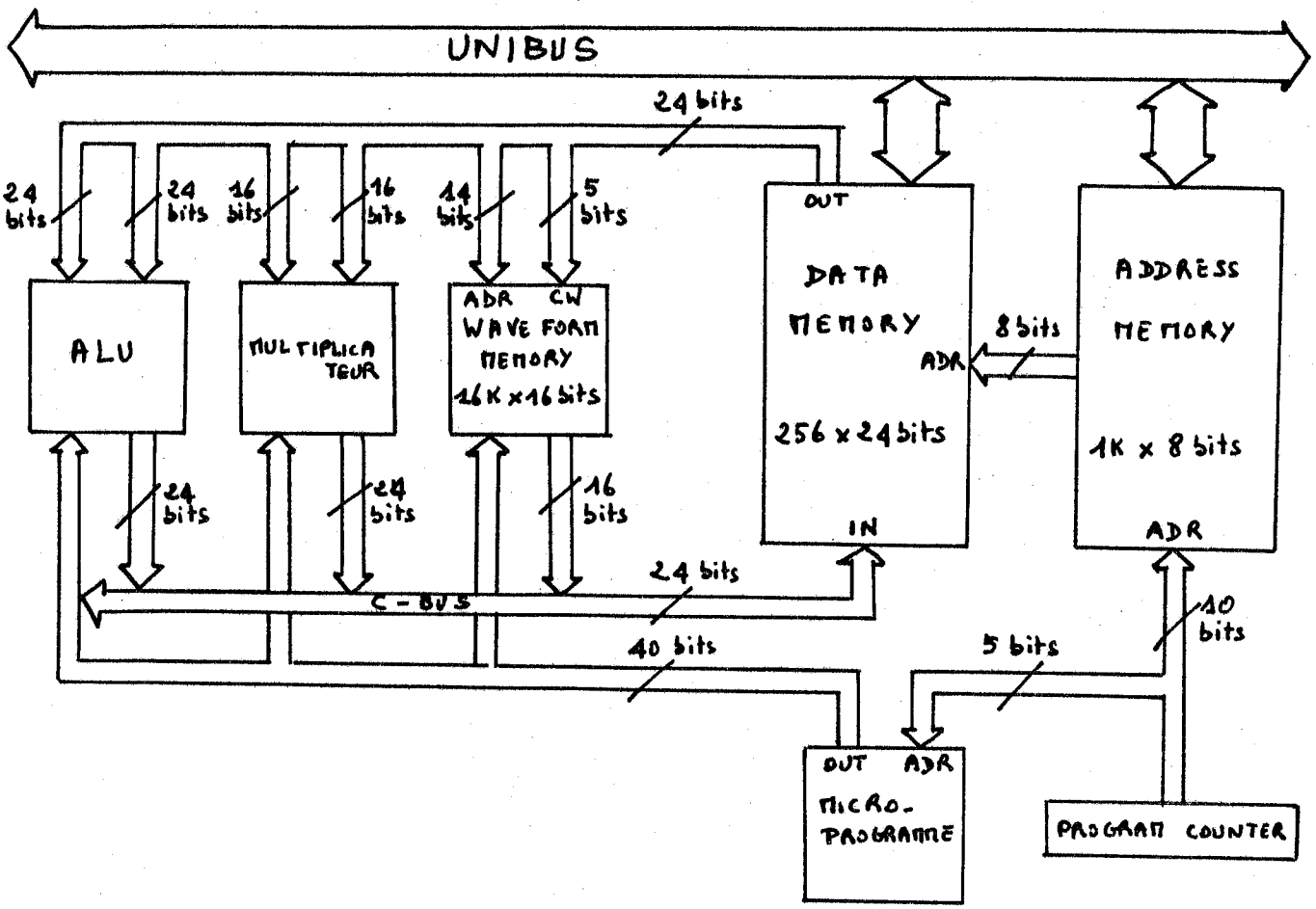

fig. 13 - unités physiques de base et bus de la 4C

produit, en complément à 2, sur 24 bits (les plus significatifs des 32 du résultat); successivement, additionne sur 24 bits une troisième opérande au produit et fournit ce résultat en sortie. Le fait de prendre systématiquement les parties les plus significatives (des

opérandes et du résultat) pour la multiplication est dû au fait que le circuit qui exécute cette opération, un MPY-16AJ de TRW **[1977]**, considère toutes les données à manipuler comme normalisées à 1, ce qui protège contre le débordement de capacité du résultat (qui est toujours inférieur aux opérandes). 2 unités par cycle intermédiaire, soit 64 unités par cycle complet.

- unité logique: elle accepte en entrée trois opérandes à 24 bits, qu'on appellera <curr>, <inc> et <fin>; la première sortie, <curw>, est obtenue comme suit:

$$\text{<curw>} = \min[(\text{<curr>}+\text{<inc>}),\text{<fin>}] \qquad \text{si } \text{<inc>} \geq 0$$
$$\text{<curw>} = \max[(\text{<curr>}+\text{<inc>}),\text{<fin>}] \qquad \text{si } \text{<inc>} < 0$$

  En même temps, <curr> est connecté en entrée à un bloc multiplicateur-accumulateur, du même type que ceux déjà décrits. 1 unité par cycle intermédiaire, soit 32 unités par cycle complet.

- timer: il accepte en entrée une opérande à 16 bits et la restitue décrémentée de 1; le décrément ne s'effectue pas à chaque cycle, mais selon une base des temps qui est programmable par un mot de contrôle à 8 bits selon la formule:

$$\text{DEC.TIM} = (n + 1) \times 0.125 \text{ msec}$$

  où n est le numéro déposé dans le mot de contrôle. Par exemple, si n = 7 les timers effectuent les décréments à intervalles de 1 msec. Chaque timer, s'il a été habilité à le faire, génère une interruption lorsque l'opérande devient négative pour dépassement de capacité (c'est à dire une décrementation après son passage à zéro). L'interruption est générée à la fin d'un cycle complet, pour éviter la désynchronisation des timers entre eux. Chaque timer qui provoque une interruption peut inscrire son numéro dans une mémoire FIFO (first-in first-out), qui sera lue ensuite par l'ordinateur maître au moment de servir l'interruption. Les habilitations à l'interruption et à l'écriture dans le FIFO se font par un mot de contrôle, propre à chaque unité. 1 unité par cycle intermédiaire, soit 32 unités par cycle complet.

La table 2 montre en détail le microprogramme du système, par lequel on réalise les unités élémentaires de fig. 10. Les numéros de cycle sont écrits en octal. Par exemple dans les microcycles "12, "13 et "14 (le signe " indique, dans le reste de cet exposé, que le nombre qui suit est écrit en octal) la ALU lit trois données à 24 bits de la Data Memory (phr-a, f1-a et f2-a) et en fait la somme. Le résultat est mis dans le C-BUS (voir fig. 13) pendant le cycle "17 (phw-a), et est récupéré par la Data Memory (à l'adresse spécifiée dans l'Address Memory). En même temps, la donnée qui transitait en M-OUT durant le cycle "12

a été mémorisée dans un registre interne, et présentée par celui-ci aux entrées d'adresse de la Waveform Memory, dont la sortie est prête sur le C-BUS pendant le cycle "25 (wt-a), d'où elle rejoint la Data Memory. On a ainsi réalisé le premier bloc de la fig. 10. Avec des considérations analogues on peut construire, à partir de la table, les autres modules.

Table 2

numéro de microcycle	opération	read/write
0	curr	R
1	inc	R
2	fin	R
3	PDP	R/W
4	cw-a	R
5	ofs	R
6	curw	W
7	cw-b	R
10	data-out	W
11	out-env	W
12	phr-a	R
13	f1-a	R
14	f2-a	R
15	m1-a	R
16	m2-a	R
17	phw-a	W
20	phr-b	R
21	f1-b	R
22	f2-b	R
23	inm-a	R
24	phw-b	W
25	wt-a	W
26	m1-b	R
27	m2-b	R
30	outm-a	W
31	timr	R
32	inm-b	R
33	timw	W
34	wt-b	W
35	fifo	W
36	outm-b	W
37	scal	R

En faisant les interconnexions entre ces modules de base on peut obtenir une infinité d'éléments de synthèse ou de traitement du signal: par exemple, en liant phw et phr (-a ou -b) d'un des modules du premier type (c'est à dire en leur donnant un même registre de la Data Memory) et en mettant à zéro l'entrée fr2, on obtient un oscillateur, dont frl représente la fréquence et wt la sortie. Si on connecte en cascade un multiplicateur, on aura aussi un contrôle de l'amplitude, avec possibilité d'accumulation de la sortie avec celle d'un ou plusieurs autres oscillateurs. Si on utilise aussi l'entrée fr2, chaque oscillateur peut étre modulé en fréquence (on peut obtenir jusqu'à 64 unités de ce

type).

Si on utilise ce dernier module uniquement pour la lecture de la forme d'onde, on obtient un dispositif de distorsion non-linéaire, ce qui permet d'utiliser la technique de synthèse par Waveshaping.

Si, toujours dans ce module, on prend la sortie à phw, on obtient un additionneur à trois entrées; avec ce module, avec les multiplicateurs et en utilisant les locations de la Data Memory comme éléments de retard, on peut simuler n'importe quel système linéaire à temps discret, éventuellement non stationnaire **[RABINER, L., GOLD, R., 1975]**; en particulier, on peut réaliser des filtres, même bi-dimensionnels et/ou variables.

L'unité logique (voir fig. 10) a été étudiée spécialement pour la construction de générateurs d'enveloppes (par exemple de fréquence et d'amplitude pour les oscillateurs), à utiliser avec les timers (avec une de ces unités et un timer on peut obtenir n'importe quelle fonction d'enveloppe, approximée par segments) [22]. Mais rien n'empêche de l'utiliser effectivement comme unité logique, et avec elle on peut réaliser plusieurs blocs non linéaires comme redresseurs de signaux, suiveurs d'enveloppe (d'un signal extérieur par exemple), détecteurs de pitch, unités échantilloneur-bloqueur (sample and hold), et ainsi de suite.

L'application principale des timers dans la 4C, comme on vient de le dire, est la construction d'enveloppes pour l'évolution des paramètres des unités de synthèse: le timer est utilisé pour indiquer à l'ordinateur les instants où il faut changer les données de ces unités (voir note [22]).

Si par exemple il faut mettre à jour un certain paramètre tous les 10 millisecondes, on pourra lui associer un timer, programmé pour un incrément chaque milliseconde, et l'initialiser avec la valeur 9 (pour qu'il devienne négatif après 10 décrementations). Chaque fois qu'il y a interruption, la routine relative changera la valeur du paramètre et réinitialisera le timer pour la prochaine mise à jour.

Toutes les unités dont on vient de parler ont été extensivement testées à l'IRCAM, et il est toujours possible, pour le compositeur qui travaille sur le système, d'en définir de nouvelles et mieux adaptées à ses propres nécessités.

Il est évident qu'il est très avantageux de pouvoir définir dynamiquement dans n'importe quel moment, à la limite pendant l'exécution, toutes ou une partie des unités du système.

Dans un cycle intermédiaire, un microcycle ("3) est réservé au PDP-11: dans ce cycle le système ne fait rien (sauf des opérations internes à la Functional Unit), mais toutes les mémoires sont multiplexées sur le bus de l'ordinateur (voir fig. 12 et 13), qui peut ainsi les lire ou bien y écrire des

nouvelles données.

Un autre microcycle (data-out, n. "10) est réservé à l'éventuelle écriture d'une donnée finale dans un de 4 registres externes, connectés à 4 DACs.

Les données sorties par l'Address Memory pendant les cycles "4 et "7 ne sont pas utilisées par le système pour adresser la Data Memory, mais sont interprétées comme des control-words spécifiant certaines des opérations à faire pendant le cycle intermédiaire actuel, comme l'habilitation à l'interruption pour le timer, le bloc de forme d'onde à adresser pour les additionneurs/lecteurs, les registres des DACs sur lesquels sortir les données, etc.

La durée d'un cycle complet étant de 62.5 usec, la fréquence d'échantillonnage de base du système est de $1/(62.5 \times 10^{-6}) = 16$ kHz; si on programme l'Address Memory de façon à ce que le deuxième groupe de 16 cycles intermédiaires soit la copie exacte du premier groupe, chaque opération sera effectuée maintenant deux fois par cycle complet, c'est à dire tous les 31.25 usec. Ceci donne une fréquence d'échantillonage double (32 kHz). Evidemment le nombre des différentes opérations réalisables se réduit de moitié, les unités du deuxième groupe étant utilisées pour doubler celles du premier. En itérant le processus on peut obtenir des taux d'échantillonnage toujours doublés, au prix d'une diminution correspondante des unités de calcul. En outre - ce qui n'est pas possible avec la 4A - on peut prédisposer des unités de calcul à 32 kHz, en doublant les interconnexions relatives, alors que d'autres peuvent continuer a marcher à la cadence minimum; d'autres encore peuvent opérer à 64 kHz (en quadruplant les interconnexions relatives), et ainsi de suite.

Toutes les mémoires de la 4C apparaissent, multiplexées, dans la mémoire centrale du PDP-11, dans une fenêtre de 1 k mots à partir de l'adresse "160000, et peuvent donc être directement manipulées par n'importe quelle instruction de l'ordinateur; la sélection de la mémoire accessible se fait par l'intermédiaire d'un control word.

L'architecture du bus du PDP-11 prévoit un adressage par bytes de la mémoire; les adresses paires correspondent à un mot de 16 bits (pour les instructions qui travaillent sur des mots), ou bien au byte le moins significatif du mot (pour les instructions qui travaillent sur des bytes); les adresses impaires correspondent au contraire au byte le plus significatif. Pour avoir la possibilité de sélectionner des mots de 24 bits (présents dans la Data Memory) à des adresses paires consécutives, la logique de décodage a été modifiée de façon que, pour chaque mot, l'adresse paire puisse se référer aux 16 bits les plus significatifs (indivisibles), les 8 derniers bits figurant dans l'adresse impaire.

APPENDICE III : Description technique de la 4X

III.1 Introduction

La 4X est un processeur multi-cartes entièrement digital spécialisé dans la synthèse et le traitement de un ou plusieurs signaux acoustiques en temps réel (jusqu'à 16 canaux d'entrée et 16 canaux de sortie); elle est capable d'engendrer plusieurs centaines d'oscillateurs, de générateurs d'enveloppe, de filtres, de modulateurs et d'autres unités dont la configuration peut être changée dynamiquement par programme.

Les chiffres donnés sont volontairement approximatifs car le nombre effectif pour chaque unité peut largement changer en fonction de la configuration hardware choisie (également variable), de la fréquence d'échantillonage choisie, et de la complexité des unités réalisées.

Dans cet exposé, comme on l'a déjà précisé (voir note [21]), on se référera systématiquement aux caractéristiques du prototype réalisé à l'IRCAM; dans le dernier paragraphe on précise les principales différences entre celui-ci et la version industrielle du processeur.

III.2 Description générale du système

Dans la 4X il y a 11 emplacements (fig. 14), pour loger des cartes qui peuvent contenir jusqu'à 220 circuits intégrés (fig. 15 et 16), habituellement en technologie TTL Shottky. Ces cartes peuvent communiquer entre elles au moyen d'un bus interne (INT-BUS - fig. 11), divisé dans la figure en trois parties.

Trois de ces emplacements sont occupés par des cartes spécialisées, chargées d'interfacer le système avec l'ordinateur qui le gère (un PDP-11/55 de la Digital Equipement Corporation) [23] et du contrôle général des unités et du bus interne. Les 8 emplacements restants peuvent être occupés par des unités d'usage général, pouvant se présenter sous des formes légèrement différentes entre elles (en ce qui concerne la Program Memory; voir plus avant), mais qui ont toutes, en particulier, le même type d'interface sur le bus interne, et qui sont donc interchangeables entre elles.

Le total des mémoires pouvant être contenues dans la 4X est de l'ordre de plusieurs centaines de k octets, mais le PDP-11 les voit toutes, multiplexées, dans un espace total de 2 k mots dans sa mémoire centrale.

Il est préférable, pour la clarté de l'exposition, de montrer d'abord en détail la structure et le fonctionnement des unités d'usage général.

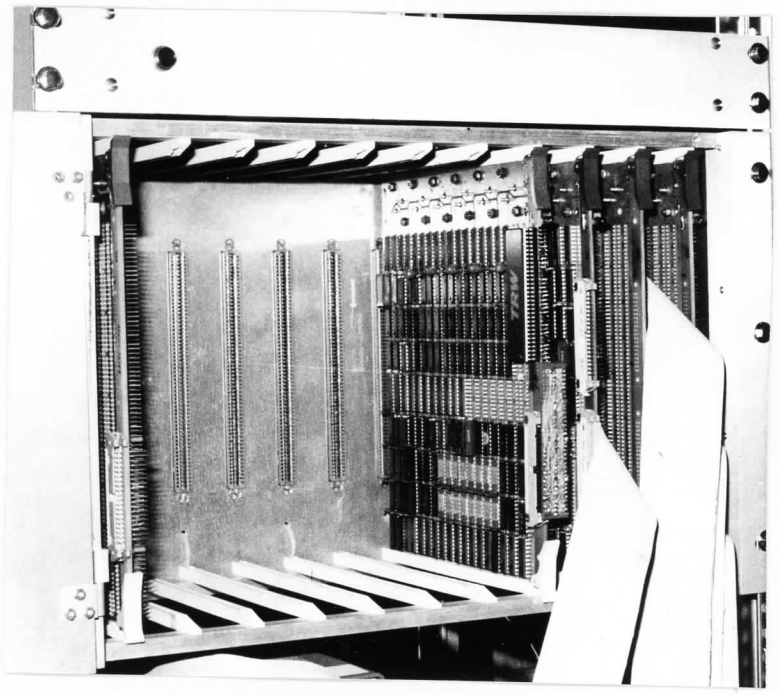

fig. 14 - vue frontale de la 4X. On peut remarquer, sur la droite, les trois cartes d'interface et de contrôle et une unité 4U microprogrammable. Le câble en premier plan lie les convertisseurs digital-analogique à l'Interconnection Unit; au dessus, le connecteur pour les ADC.

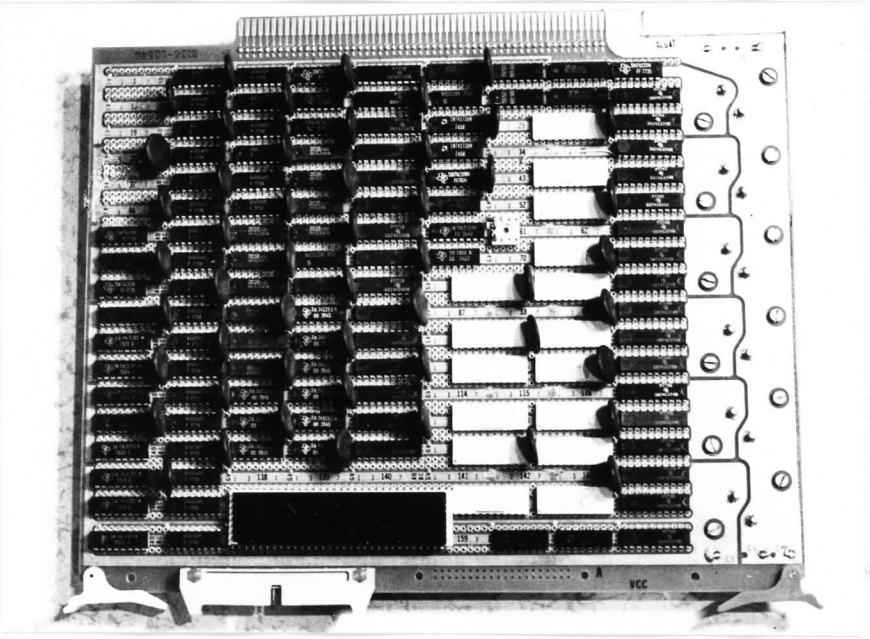

fig. 15 - une carte de la 4X. Côté circuits.

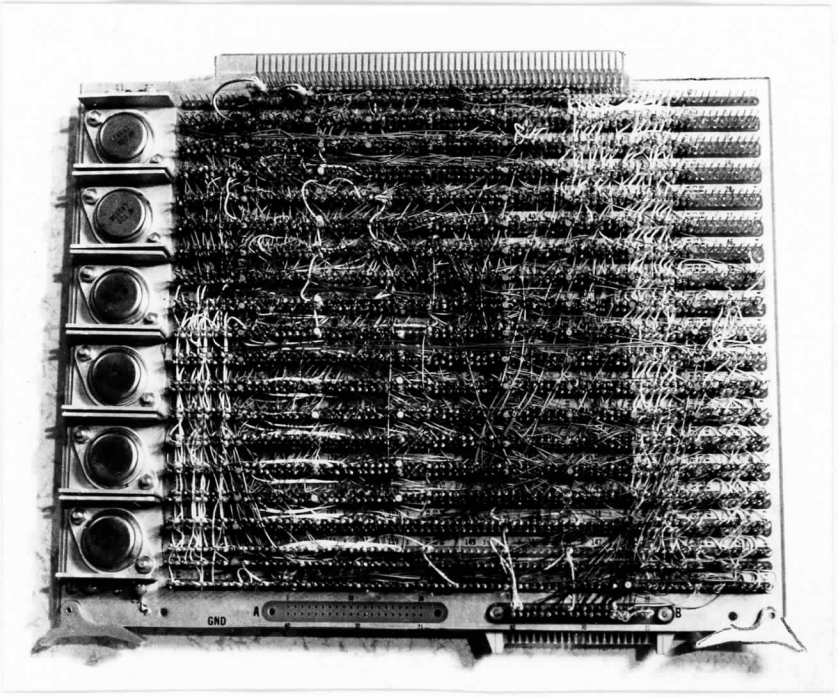

fig. 16 - une carte de la 4X. Côté cablage.

III.3 Architecture des unités de synthèse

L' unité de base destinée à la synthèse ou au traitement du signal, appelée 4U, présente une architecture du type montré en fig. 17; son noyau central est constitué par trois mémoires (Data Memory, Address Memory et Program Memory), et par une unité de calcul (Functional Unit). Comme on le voit, au moins à un niveau global, il s'agit essentiellement du même type de structure de la 4C (voir Appendice II).

La Functional Unit de la 4U est montrée plus en détail fig. 18. Les éléments essentiels qui la composent sont: un multiplicateur, une unité logique-arithmetique (ALU) utilisable pour additions, soustractions et opérations logiques, une Waveform Memory et un registre à décalage (SHIFT REG).

L'élément déterminant, qui donne une extrême polyvalence à la structure, est de toute façon l'ensemble des bus internes et des registres, de pipe-line et d'interconnexion, qui complètent le tout. Ceux-ci ont été étudiés pour permettre un maximum de liberté dans la conception du microprogramme, qui, en établissant la succession temporelle des signaux de clock et de strobe des registres, définit en pratique le type et la quantité des opérations exécutables. Il optimise donc les ressources caractéristiques de l'unité selon le type d'application. En fait, établir à quoi est destinée spécialement une 4U a peu de sens si l'on n'a pas établi tout d'abord un microprogramme.

Le microcycle de base de chaque unité [24] est de 61.035 nsec, obtenu à partir d'un Master Clock fonctionnant à 16.384

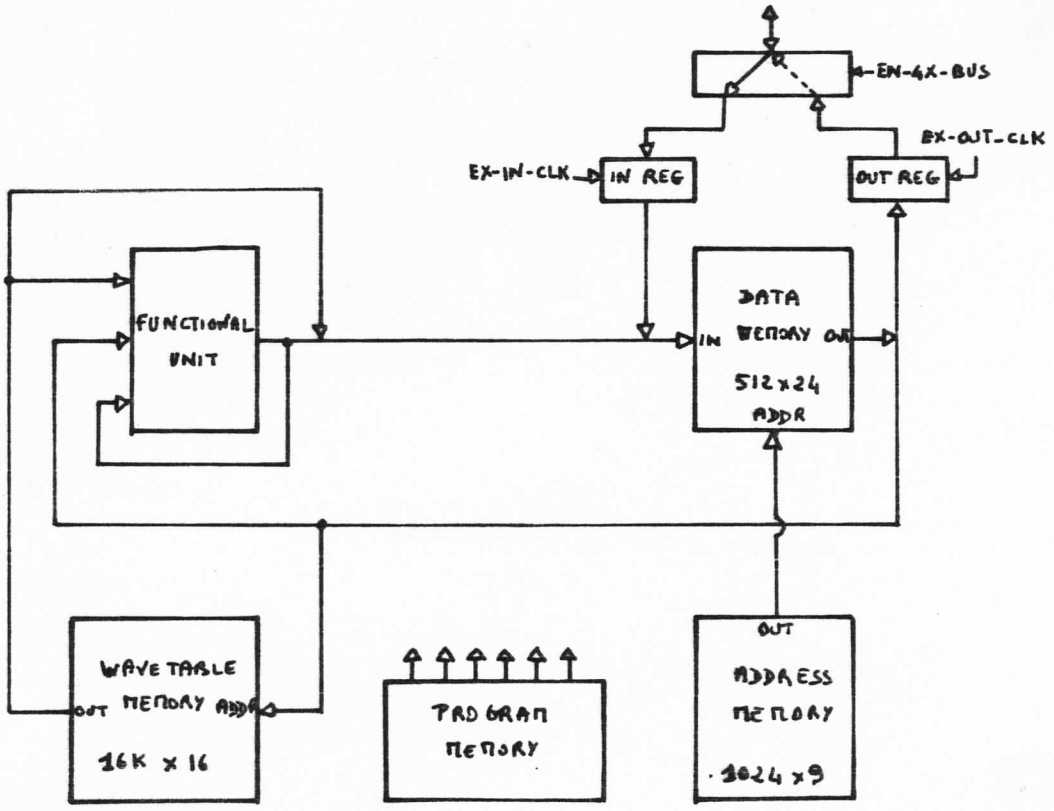

fig. 17 - architecture des unités de synthèse 4U

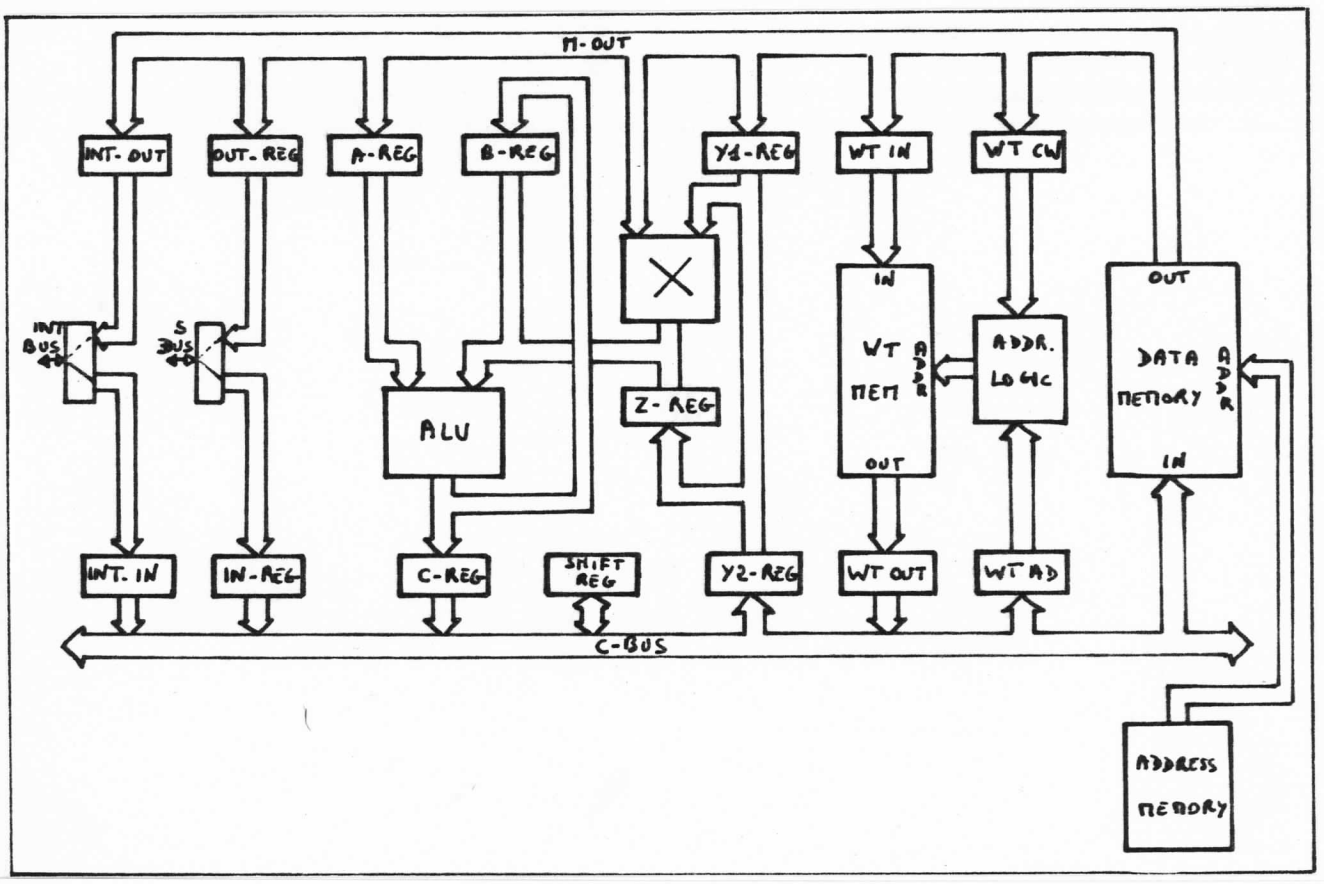

fig. 18 - structure physique des unités de synthèse 4U

MHz. Dans chaque microcycle l'Address Memory, adressée cycliquement par un compteur, sélectionne un des registres de la Data Memory. En conformité avec le microprogramme, la Functional Unit pourra soit écrire dans le registre sélectionné le résultat d'une opération effectuée dans les cycles précédents, soit en prendre une donnée sur laquelle opérer dans les cycles suivants. En outre, la Functional Unit exécute une opération élémentaire ou seulement une partie d'elle. Tout cela se déroule en parallèle, toute la structure étant en pipe-line. Un cycle complet de chaque unité est constitué évidemment par 1024 microcycles, chiffre égal au nombre de locations de mémoire de l'Address Memory. Ensuite les mêmes opérations sont répétées identiquement sur les mêmes opérandes (identifiés par les adresses écrites dans l'Address Memory).

Le mécanisme de définition des modules et des interconnexions est donc, comme on le voit, identique à celui décrit à propos de la 4C (voir Appendice II).

La Program Memory, qui est responsable des unités élémentaires de la carte, est une mémoire à mots de 32 bits existant actuellement en trois versions différentes: l'option de base, reprise de la 4C, est une ROM à 32 locations de mémoire, adressées cycliquement comme pour l'Address Memory. Le microprogramme par conséquent est constitué d'une séquence de 32 micro-instructions, qui se repète 32 fois dans un cycle complet. Ce cycle est donc subdivisé en 32 cycles intermédiaires, chacun de la durée de 2 usec environ, équivalents entre eux au point de vue des opérations qui sont effectuées. Naturellement ces opérations ne sont pas les mêmes, car elles agissent en général sur des opérandes différents et placent les résultats dans des locations de mémoire différentes.

Dans un cycle de 2 usec il y a des microcycles prévus pour l'échange des données entre unités: un ou deux cycles d'entrée des données venant du bus interne, et un nombre égal de cycles de sortie. En correspondance avec ces cycles, les signaux EN-4X-BUS et EX-IN-CLK (fig. 17), appartenant au CTRL INT-BUS (montré fig. 11), sont engendrés par une des unités de contrôle (voir plus loin) et la donnée correspondante à l'adresse mémorisée dans l'Address Memory est envoyée, à travers OUT REG, sur le DATA INT-BUS ou bien elle est lue, à travers IN REG (le signal EX-OUT-CLK est par contre engendré à l'intérieur de chaque unité et il est toujours présent, indépendamment du fait que la donnée en question soit habilitée ou non à se présenter sur le bus). Un autre microcycle est réservé systématiquement au PDP-11: dans ce cycle l'unité ne fait rien (sauf des opérations internes à la Functional Unit), mais toutes les mémoires sont multiplexées sur le bus de l'ordinateur (non indiqué, par simplicité, en fig. 17: voir la fig. 11), qui peut ainsi les lire ou bien y écrire des nouvelles données.

Dans la deuxième option disponible, la Program Memory se présente comme une mémoire de 512 mots de 32 bits, divisée en 16 microprogrammes de 32 mots chacun, qu'on peut sélectionner par l'ordinateur maître. De cette façon, il est même possible de changer de microprogramme en temps réel, et d'obtenir des prestations totalement différentes avec une même carte, en reconfigurant ainsi dynamiquement tout le système.

Dans la troisième option, enfin, la Program Memory est une RAM de 1024 mots (toujours à 32 bits), accessible par l'ordinateur maître; cela permet, selon la façon dont elle est programmée, d'obtenir des cycles intermédiaires de longueurs différentes: à la limite, le cycle total peut se composer d'une séquence de microcycles tous différents entre eux, et le cycle intermédiaire coïncide dans ce cas avec le cycle total. Cela aussi, naturellement, peut se faire en temps réel.

Dans toutes les 4U, les Program Memories se trouvent physiquement sur une plaquette qui est reliée à la carte par le moyen d'un connecteur (fig. 19); il est donc possible de changer d'option à tout moment (par exemple avant chaque session en studio, ou avant un concert en direct). Plus loin, nous verrons quelques exemples d'utilisation de ces caractéristiques.

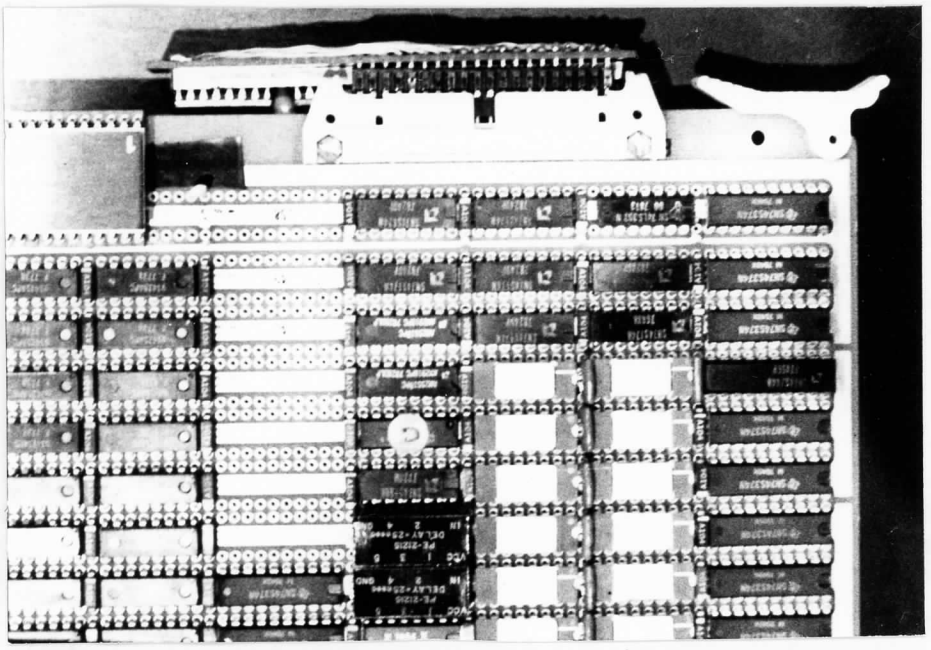

fig. 19 - détail d'une carte, qui montre la plaquette contenant la Program Memory d'une unité 4U, insérée dans le connecteur; en haut à gauche on peut voir un des chips composant la ROM.

Il faut ajouter que dans la fig. 17 manque une dernière chose: le WTAD INT-BUS, qui adresse la Waveform Memory.

Les données manipulées par les unités des 4U sont typiquement des mots de 24 bits; il y a deux exceptions, qui sont les mêmes que pour la 4C: les lectures de Waveform Memory, qui donnent 16 bits, et les multiplicateurs, qui effectuent le calcul sur 16 bits et donnent 24 bits sur les 32 du résultat.

Un autre aspect à souligner est la possibilité de changer le taux d'échantillonage, selon un méchanisme parfaitement identique à celui décrit à propos de la 4C (voir Appendice II). Même pour la 4X, donc, 16 kHz représentent seulement la valeur minimum des taux d'échantillonage utilisables.

Comme on l'a déjà dit, les cartes 4U sont en fait une évolution du synthétiseur 4C; leur architecture détaillée est montrée fig. 18. Par rapport à la 4C, telle qu'on l'a décrite dans l'Appendice II et montrée fig. 13, les unités 4U présentent en total les différences suivantes:

- la taille de la Data Memory est de 512 registres à 24 bits (contre 256 pour la 4C);

- en conséquence du premier point, la largeur de mot de l'Address Memory a été portée à 9 bits ($512 = 2^9$);

- il est possible de programmer l'opération des unités arithmétiques-logiques (ALUs), alors que dans la 4C elles effectuent exclusivement des additions (au moins d'un point de vue externe; en réalité, la 4C utilise la ALU pour faire aussi des soustractions, mais cela est interne au fonctionnement des unités logiques);

- un registre à décalage programmable a été ajouté (sur le C-BUS; voir fig. 18);

- il est possible d'écrire dans la Waveform Memory directement à partir de la Data Memory (dans la 4C, on peut seulement lire cette mémoire; seul l'ordinateur maître peut écrire dedans);

- dans la 4C, comme on l'a déjà dit, pendant les cycles cw-a et cw-b (voir table 2) la sortie de l'Address Memory n'est pas utilisée pour adresser la Data Memory, mais est interprétée directement comme un control-word; dans la 4U, la Data Memory est néanmoins adressée, et c'est son contenu qui est pris comme control-word. L'avantage de cette technique est que maintenant on dispose de 24 bits pour les mots de contrôle, contre 8 bits dans le cas de la 4C;

- il n'y a aucune possibilité de multiplexer les Waveform Memories dans l'espace de travail de l'ordinateur: ces mémoires sont exclusivement accessibles par des registres de l'Interface Unit (voir plus loin);

- la Waveform Memory reçoit les adresses par le C-BUS, et non plus par M-OUT; cela entraîne, entre autre, que la première unité élémentaire montrée fig. 10 est maintenant réalisée sous la forme, légèrement différente, de la fig. 20;

- le multiplicateur peut prendre une opérande à partir du C-BUS (et non seulement à partir de M-OUT);

- il n'y a pas de timers dans les 4U: ils sont tous regroupés dans une des cartes de contrôle (Interconnection Unit);

- la Data Memory peut communiquer, évidemment, non seulement avec le bus de l'ordinateur maître (S-BUS), mais aussi avec le INT-BUS pour échanger des données avec les autres 4U.

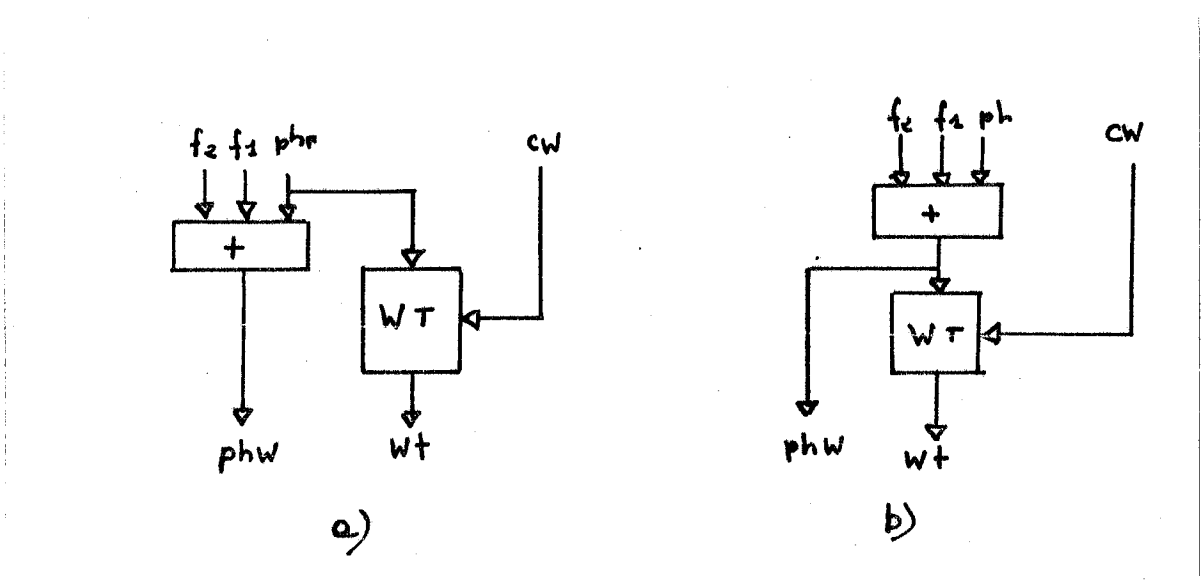

fig. 20 - réalisation de l'unité additionneur - lecteur de tableau dans la 4C (a) et dans la 4U (b)

III.4 Architecture des unités de contrôle

Après avoir éclairci la structure des unités générales, nous allons passer à l'examen approfondi des fonctions développées par la carte d'interface et les deux cartes de contrôle du système.

La carte d'interface, c'est à dire la Bus Terminator Unit, réalise simplement, dans le prototype, l'extension bidirectionnelle des 44 signaux de l'UNIBUS (bus du PDP-11) vers les cartes de la 4X, sans surcharger électriquement les drivers de l'ordinateur. Cette extension est montrée, en fig. 11, sous le nom de S-BUS [25].

La première des deux cartes de contrôle, l'Interface Unit, comprend les unités suivantes:

- 16 registres d'interface à 16 bits;
- logique de dialogue avec l'ordinateur;
- logique de sélection des mémoires (Memory Management);
- logique d'adressage progressif des Waveform Memories;
- un buffer (mémoire tampon) de 16 k mots, et la logique de commande relative.

Les registres d'interface constituent, pour le PDP-11, la voie usuelle de dialogue avec le système. Ils apparaissent toujours dans la mémoire centrale de l'ordinateur, à partir de l'adresse "164000. La figure 21 montre synthétiquement les fonctions des différents registres.

Les bits du premier registre, R0, sont divisés en 8 champs; les deux premiers servent pour le contrôle de l'accès au bus du PDP-11 par les mémoires des unités de synthèse (Memory Management). L'information déposée en UNIT SELECT et MEMORY SELECT est décodée, et un signal sur le CTRL INT-BUS habilite une seule mémoire du système sur le S-BUS; la valeur déposée en UNIT SELECT permet de sélectionner une des 8 unités de synthèse (numérotées de 0 à 7) ou l'Interconnection Unit (pour son Interconnection Memory: voir plus loin); dans le cas des unités de synthèse, si MEMORY SELECT = 0 la Data Memory est sélectionnée; si MEMORY SELECT = 1 l'Address Memory est sélectionnée.

La mémoire ainsi sélectionnée apparaît dans la mémoire centrale du PDP-11, dans une fenêtre de 1 k mots, à partir de l'adresse "160000. Les données qu'elle contient peuvent alors être manipulées par n'importe quelle instruction de l'ordinateur.

Les signaux de contrôle de l'UNIBUS nécessaires pour le dialogue avec l'ordinateur dans les opérations de lecture ou d'écriture sont engendrés à l'intérieur de l'Interface Unit, qui est chargée de faire cela pour tout le système. La logique existant sur chaque carte séparément est seulement chargée d'envoyer ou de recevoir les données sur le S-BUS.

Le dernier champ de R0 est un interrupteur de contrôle de tout le système; le septième (ONE STEP) permet de faire marcher la 4X en pas-à-pas, ce qui est très utile surtout dans la phase de mise au point du système; le troisième champ habilite ou non une interruption en mode pas-à-pas.

Le sixième champ de R0, et le registre R1, servent pour l'adressage de la Waveform Memory: WT ADDRESS est initialisé avec la première adresse dans laquelle on veut écrire, et en WT AD

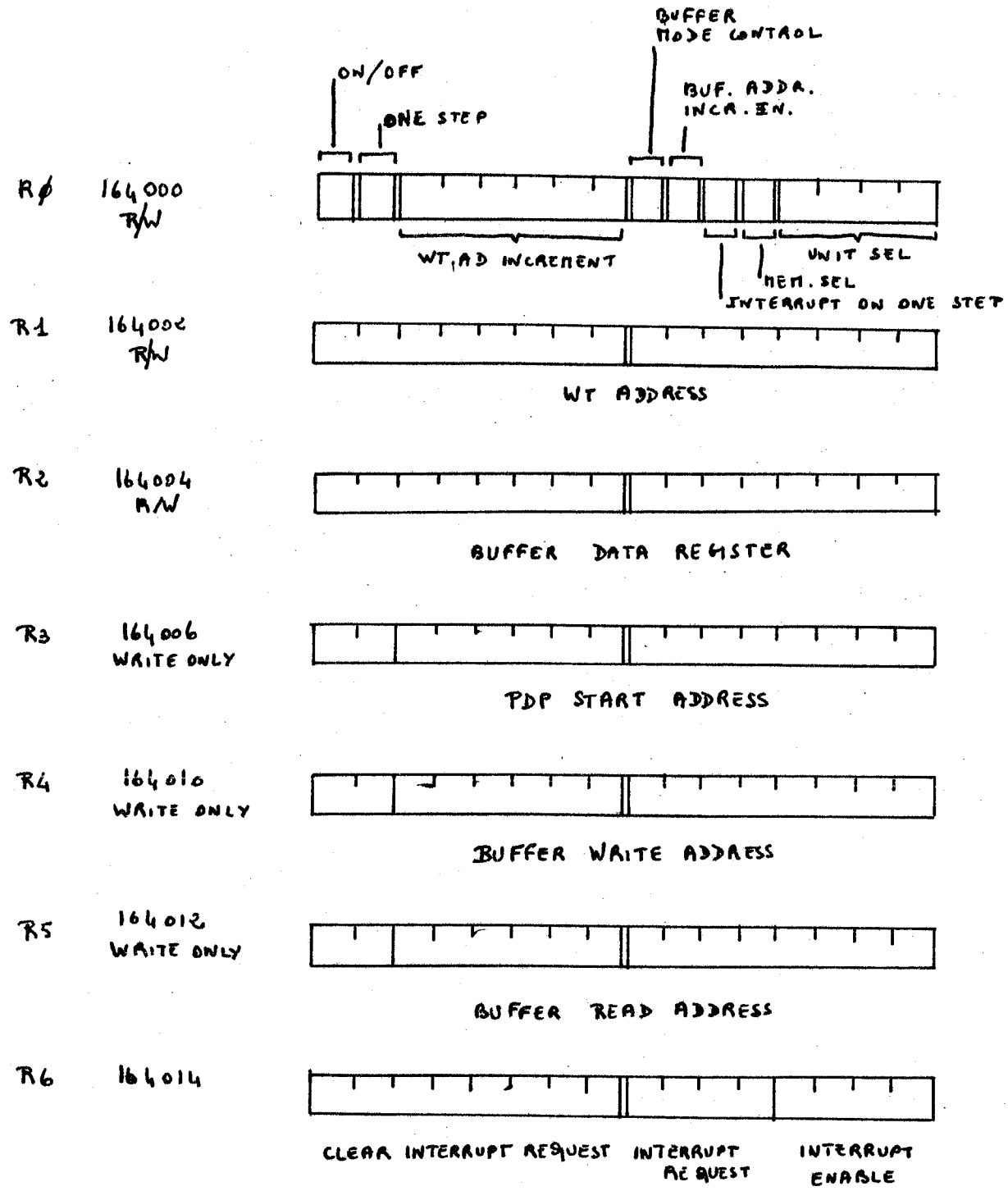

fig. 21 - les premiers registres de l'Interface Unit.

INCREMENT on spécifie un éventuel pas d'adressage de la mémoire. Le contenu de R1 est présenté sur le bus interne (WTAD INT-BUS, fig. 11) et il va adresser les huit mémoires. Ensuite il est possible d'écrire (ou de lire) dans la mémoire choisie: chacune d'elles se présente à l'ordinateur comme un registre, aux

adresses "164020-"164036 (registres R8 - R15). Chaque accès à une de ces adresses provoque, automatiquement, une accumulation en R1 de la valeur déposée en WT AD INCREMENT.

Le chargement des formes d'onde peut être fait en DMA, pendant que l'ordinateur continue à dialoguer avec le système. Un transfert de toute une mémoire de 16 k mots entre la 4X et un disque type RK06 se fait en environ 64 msec. (avec un PDP-11/55).

Finalement, les registres restants (de R2 à R6), sont dédiés à la gestion du buffer. Cette mémoire, constituée par 16 k mots de RAM, peut être d'usage général, mais elle a été conçue surtout pour des applications en DMA (par exemple pour le transfert rapide du disque au système des signaux mémorisés précédemment et à traiter en temps réel, et vice versa).

Le buffer, comme les Waveform Memories, se présente à l'ordinateur comme un seul registre de 16 bits, R2, dans lequel on effectue les lectures ou écritures en séquence. Pour l'adressage il y a trois pointeurs: un pour l'accès de l'ordinateur (R3), et deux pour l'accès du système (R4 pour l'écriture par le DATA INT-BUS au buffer, R5 pour la lecture en sens invers); l'ordinateur utilise un seul registre pour les lectures et les écritures, car il peut effectuer une seule de ces opérations à la fois, tandis que la 4X en utilise deux, car elle est capable de lire et d'écrire simultanément (plus exactement, elle arrive à faire les deux choses dans un seul cycle complet). Les trois pointeurs sont incrémentés de 1 à chaque lecture ou écriture qui les concerne, de telle façon que l'accès en séquence par le PDP-11 soit automatique, à travers le registre R2.

Les index sont à 14 bits, pour adresser tous les 16 k mots. Habituellement R4 a le bit le plus significatif à 1 et R5 à 0; ils sont incrémentés modulo 8 k locations et chacun peut adresser une moitié de la mémoire totale. Cela signifie que la 4X lit normalement la première moitié du buffer, et écrit dans la deuxième; l'ordinateur fera naturellement le contraire. Ce dernier (ou n'importe quel autre dispositif capable d'engendrer un transfert en DMA, par exemple un disque) dépose l'adresse initiale du buffer en R3 (normalement, 0 pour les lectures et 8192, c'est à dire le début de la deuxième moitié de la mémoire, pour les écritures), et envoie (ou reçoit) en séquence les données dans R2. Le bit le plus significatif de R3 n'est pas modifié par les incréments, et il reste toujours fixé au moment de l'initialisation: par conséquent R3 aussi pointe au buffer modulo 8 k; on continue à lire dans la première moitié du buffer et à écrire dans la deuxième.

On peut diviser chacune de ces unités en deux blocs de 4 k mots, et, chaque fois que R4 ou R5 arrivent à la fin d'un bloc, une interruption est engendrée, pour informer l'ordinateur que le bloc est libre à nouveau (pour une nouvelle lecture ou écriture). La 4X continue en même temps à travailler sur l'autre bloc.

On a ainsi une unité de transfert rapide bidirectionnelle, fondée sur la technique classique du double buffer réalisée en hardware. Pour chacun des quatre blocs est prévue une routine d'interruption différente (vecteurs d'interruption aux adresses "440, "444, "450, "454), de manière à ce que ces dernières puissent être extrêmement rapides, car elles n'ont pas besoin d'examiner la situation actuelle avant de démarrer. On peut habiliter chacune de ces quatre interruptions séparément par les bits du premier champ de R6; le deuxième champ indique les éventuelles interruptions demandées mais non habilitées. N'importe quelle écriture au byte le plus significatif de R6, par exemple CLRB 6#164015, provoque la mise à zéro des deux premiers champs. Finalement, le bit de BUFFER MODE CONTROL (en R0) permet, lorsque il est levé, d'utiliser R3, R4 et R5 comme des pointeurs normaux à 14 bits, afin d'adresser tout le buffer dans des applications autres que celle que nous venons de décrire, alors que le bit de BUFFER ADR INCR EN peut, au contraire, déshabiliter totalement l'incrément des pointeurs, si bien que toutes les lectures et les écritures se font toujours à la même adresse.

Le registre R7 est reservé pour les developpements futurs du système.

La deuxième carte de contrôle, soit l'Interconnection Unit, comprend les unités suivantes:

- logique de commande pour les interconnexions entre unités de synthèse;
- 8 blocs de 32 timers chacun;
- logique de préparation et génération des interruptions associées aux timers;
- logique de gestion des convertisseurs (DACs et ADCs).

La première de ces unités est constituée principalement d'une mémoire de 512 mots de 8 bits (Interconnection Memory). Chaque mot spécifie une interconnexion: il se divise en fait en deux champs de 4 bits chacun, destinés tous les deux à contenir l'adresse d'une unité (d'une façon analogue à UNIT SELECT du registre R0; voir fig. 21). L'un de ces champs représente l'unité d'origine, c'est à dire celle qui envoie une donnée à l'autre; le deuxième représente l'unité de destination, qui donc reçoit la donnée.

Comme l'Address Memory, l'Interconnection Memory est adressée par un compteur, mais avec un incrément tous les deux microcycles de base: un cycle d'interconnexion dure donc 120 nsec environ. Un cycle complet de la 4X est formé par 512 cycles d'interconnexion.

Durant un cycle d'interconnexion, la mémoire est lue, les deux adresses (origine-destination) sont décodées et deux signaux

sont engendrés et envoyés sur CTRL INT-BUS: l'EN-4X-BUS relatif à l'unité d'origine (voir fig. 17) et l'EX-IN-CLK relatif à l'unité de destination. De cette façon, la première unité met dans DATA INT-BUS la donnée à transférer (sur 24 bits), qui sera présente dans son registre de sortie (OUT REG, fig. 17), et la deuxième unité la recueille à travers son registre d'entrée (IN REG). L'Interconnection Memory réalise donc des interconnexions entre des éléments de calcul, à l'intérieur d'un cycle total, de la même façon que les Address Memories des 4U. La seule différence est qu'il s'agit ici d'interconnexions entre différentes unités de synthèse. L'Interconnection Memory est normalement accessible par l'ordinateur à travers le Memory Management déjà décrit.

L'Interconnection Unit contient une série de registres (montrés fig. 22), dont certains sont à 16 bits et d'autres à 24 bits. Ils apparaissent toujours dans la mémoire centrale du PDP-11 aux adresses "166000 - "170777; ils sont utilisés pour la réalisation de 8 blocs de 32 timers chacun et de la logique d'interruption associée avec eux.

Un timer de la 4X est constitué essentiellement par un registre à 16 bits qui est incrémenté de 1 par intervalles de temps programmables (à la différence des timers de la 4C, qui sont par contre décrémentés). Lorsque un ou plusieurs timers d'un bloc donné arrivent à zéro pour dépassement de capacité, une interruption est engendrée (si elle avait été habilitée). L'application principale des timers de la 4X est la même que pour la 4C, déjà décrite dans l'Appendice II. Chaque bloc de timers est normalement affecté à une 4U.

A chacun des 8 blocs est associée une mémoire de type FIFO à 32 mots; ces mémoires servent pour indiquer à la routine d'interruption les timers du bloc qui sont arrivés à 0. Le vecteur d'interruption, unique pour les 8 blocs, est à l'adresse "400; les blocs ont un ordre de priorité pour la génération des interruptions (du bloc 0 - le plus prioritaire - au bloc 7).

La logique, similaire à celle utilisée dans la 4C, est la suivante: chaque timer qui arrive met son propre numéro ordinal (de 0 à 31) dans le FIFO du bloc auquel il appartient; le premier, en plus, lève un flag pour mettre en marche la logique de préparation de l'interruption. A la fin du cycle total, quand tous les timers de chaque bloc ont été incrémentés (c'est à dire quand les écritures dans les FIFO ont été complétées), le bloc le plus prioritaire de ceux qui ont au moins un timer arrivé génère une interruption. Ainsi la routine peut lire dans le FIFO les numéros des timers arrivés, et agir en conséquence. Au moment où l'interruption est engendrée tous les timers sont automatiquement bloqués, et doivent être réhabilités par software pour pouvoir recommencer à s'incrémenter, ce qui est fait normalement à la fin de la routine d'interruption; ceci est nécessaire, surtout dans les routines d'interruption très longues, pour s'assurer que les timers du bloc qu'on sert ne perdent pas d'incréments par rapport aux autres, et empêcher ainsi une désynchronisation du résultat

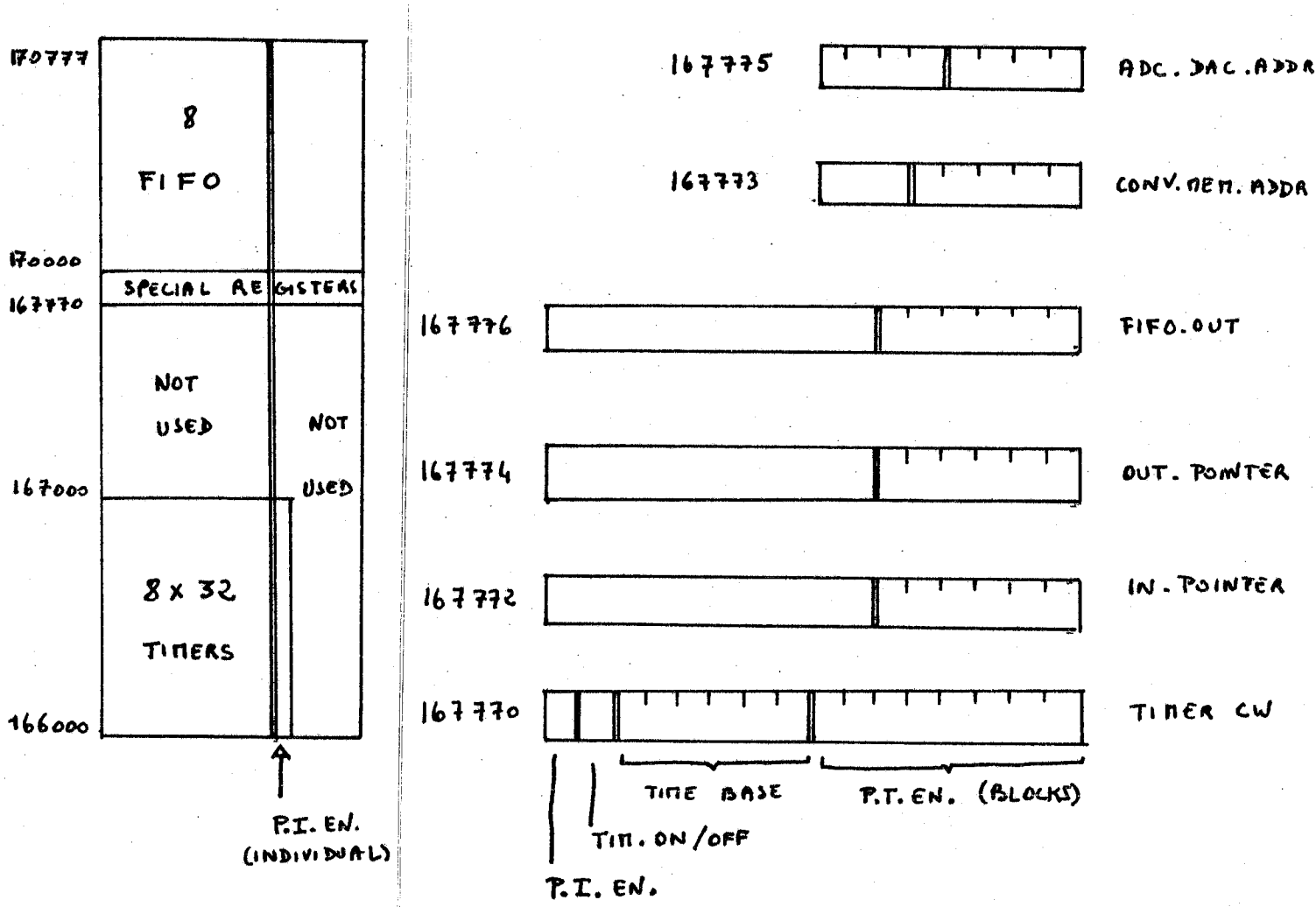

fig. 22 - les registres associés aux timers et à la logique d'interruption dans l'Interconnection Unit.

sonore produit.

On vient de dire que les registres des timers sont à 16 bits, et correspondent indivisiblement aux adresses paires (selon

la logique évoquée à propos des Data Memories: voir Appendice II). En plus on utilise un 17-ème bit (un des 8 correspondant à l'adresse impaire) pour chaque timer. Avec ce bit on habilite le timer à engendrer des interruptions et à écrire dans le FIFO.

Les 256 timers occupent les adresses "166000 - "166777 (voir fig. 22); les 8 FIFO occupent les adresses "170000 - "170777; le bloc de mémoire intermédiaire, aux adresses "167000 - "167777, n'est utilisé pour l'instant qu'en une petite partie, le reste étant réservé pour des développements futurs du système: aux dernières adresses de ce bloc il y a des registres de contrôle, à 16 et à 8 bits, comme il est montré dans la figure.

Le premier ("167770) est en pratique un mot de contrôle (Timer Control Word) des timers: le premier champ (8 bits) habilite la génération des interruptions, séparément pour chaque bloc; le deuxième fixe, sur 6 bits, la base des temps pour les timers, c'est à dire l'intervalle de temps entre deux incréments successifs. Ceci est donné par la formule:

$$DEC.TIM = (n + 1) \times 250 \quad usec$$

dans laquelle n est le numéro déposé dans ce champ; les deux derniers bits règlent, respectivement, l'habilitation à l'incrément et à la génération des interruptions de tous les timers.

Les registres IN-POINTER, OUT-POINTER et FIFO OUT (adresses "167772, "167774, "167776) servent pour l'utilisation pratique des mémoires FIFO: les deux premiers font fonction de pointeurs d'entrée et de sortie pour le FIFO du bloc qui a provoqué l'interruption, alors que la sortie de cette mémoire apparaît dans FIFO OUT multipliée par 2 (ceci est utile, pour transformer le numéro de timer arrivé en offset pour chercher, dans une table d'adresses du programme, les opérations à effectuer). Chaque lecture de FIFO OUT incrémente le pointeur de sortie du FIFO sélectionné, de manière qu'il puisse être lu avec des lectures consécutives de FIFO OUT.

Les registres à 8 bits des adresses "167773 et "167775 sont utilisés pour la gestion des convertisseurs (dont nous parlerons plus tard), et ils apparaissent à ces adresses seulement par commodité de construction.

En conclusion, les routines d'interruption des timers doivent être structurées selon les pas suivants:

- bloquer (éventuellement) tous les paramètres dont on veut conserver le synchronisme avec les timers, comme par exemple les unités logiques utilisées comme générateurs d'enveloppe; les timers, comme on l'a déjà dit, sont tous bloqués par hardware à l'entrée de chaque interruption;

- lire le pointeur d'entrée des FIFO: son contenu indiquera combien de timers sont arrivés dans le FIFO du bloc qui a provoqué l'interruption;

- lire le FIFO OUT, qui donnera le numéro du premier timer arrivé, déjà transformé en offset;

- à partir de cette donnée, faire ce qu'il faut pour la mise à jour des paramètres de la 4X;

- réinitialiser le timer avec la prochaine valeur temporelle;

- effectuer une boucle à partir du troisième pas, répétée un nombre de fois égal à la valeur lue au deuxième pas;

- remettre à zéro les deux pointeurs (d'entrée et de sortie), pour les initialiser pour la prochaine interruption;

- lever le bit TIM ON/OFF pour faire repartir tous les timers simultanément (ce bit est mis à zéro à chaque fois qu'une interruption est acceptée, pour réfléter la situation de blocage de tous les timers); réhabiliter éventuellement d'autres paramètres bloqués au début de l'interruption;

- lever le bit PI EN pour habiliter à nouveau les interruptions; ce bit, tout comme TIM ON/OFF, est mis à zéro à l'entrée de l'interruption, pour empêcher la superposition des routines (on peut servir seulement un bloc à la fois);

- restituer le contrôle au background.

L'Interconnection Unit est capable d'échanger des données avec d'autres cartes comme n'importe quelle unité de synthèse, avec le même système qu'on a décrit plus haut. Dans ce cas-là les données reçues ou envoyées ne sont pas employées comme dans les autres unités, mais sont échangées avec les convertisseurs. De cette façon il est possible d'envoyer le résultat final d'un processus de synthèse à un des DAC en passant par la Data Memory de l'unité, le INT-BUS, et l'Interconnection Unit. Si on fait le chemin à l'envers (en utilisant des ADC), on peut traiter dans la 4X un ou plusieurs signaux extérieurs.

Comme la 4X est capable de gérer jusqu'à 16 DAC et 16 ADC, on peut avoir en temps réel 16 sorties différentes; celles-ci pourront être le résultat d'un traitement (toujours en temps réel) d'un maximum de 16 signaux extérieurs (concrets ou provenants par exemple d'un autre système de synthèse, en dialogue avec la 4X).

Pour adresser les 32 convertisseurs du système, il y a dans l'Interconnection Unit une mémoire à 32 mots de 8 bits (Converter Memory), qui a, dans cette unité, un rôle parfaitement analogue à celui des Address Memories des unités de synthèse: elle établit à quelle adresse il faut envoyer les données reçues par l'IN REG

(voir la fig. 17, pour ce qui concerne l'accès au DATA INT-BUS), ou bien à quelle adresse il faut chercher les données à envoyer à l'OUT REG (évidemment l'adresse spécifie un convertisseur). Chaque mot de mémoire est divisé en deux champs, dont l'un indique le DAC et l'autre l'ADC à sélectionner dans le cycle en cours.

Dans chaque cycle de 2 usec, l'unité est capable de recevoir une donnée du INT-BUS et d'en envoyer une autre: pour faire cela, la mémoire est normalement adressée par un compteur, avec un incrément toutes les 2 usec, qui permet de sélectionner deux convertisseurs (un d'entrée et un de sortie) à brancher sur le bus. Normalement l'adresse d'un convertisseur donné est écrite une seule fois dans la Converter Memory. Cela donne sa sélection toutes les 62.5 usec, c'est à dire un fonctionnement à une fréquence d'échantillonnage de 16 kHz; mais naturellement, comme dans le cas de l'Address Memory, on peut écrire plusieurs fois l'adresse d'un convertisseur dans la mémoire pour avoir des taux d'échantillonage multiples de 16 kHz.

Pour programmer la Converter Memory il y a deux registres à 8 bits, dont on a déjà parlé, aux adresses "167773 et "167775 (montrés fig. 22). Avec le premier on adresse la mémoire, et avec le deuxième on peut écrire dans le registre ainsi sélectionné les adresses des deux convertisseurs.

III.5 Microprogrammes et applications

Pour montrer de façon plus directe les principales possibilités de la 4X, dans ce paragraphe on détaillera quatre microprogrammes "type" à 32 microcycles applicables aux unités 4U, en décrivant les modules de base qu'on peut en obtenir et les applications possibles.

On a déjà dit que l'élément qui définit vraiment une 4U en pratique est le microprogramme: selon les caractéristiques, plus ou moins spécialisées, de celui-ci, on aura des modules de base tout à fait différents entre eux, même avec une uniformité totale d'architecture.

Etant donné la complète interchangeabilité des les cartes de synthèse du système, il est clair que, en fonction des exigences spécifiques du compositeur, on pourra choisir les microprogrammes les mieux adaptées à chaque utilisation: non seulement le choix d'un microprogramme spécialise les fonctions d'une carte 4U, mais il redéfinit entièrement les modules matériels qui composent la carte en question; c'est dans ce sens que l'on parlait, dans l'introduction à l'Appendice, de configuration hardware variable.

Le premier de ces microprogrammes est montré dans la table 3: il permet de réaliser, en 2 usec, les modules représentés fig. 23. Ce microprogramme rend la 4U pratiquement identique à un synthétiseur 4C, avec quelques améliorations dont

on a déjà parlé; d'un point de vue utilitaire, on remarquera surtout que les unités logiques sont séparées des multiplicateurs, ce qui permet une utilisation plus rationnelle de ces derniers, et que les additionneurs peuvent maintenant soustraire: ce qui évite d'utiliser banalement des multiplicateurs pour effectuer des multiplications par -1, comme c'est le cas dans la 4C pour faire des soustractions.

Table 3

numéro de microcycle	opération	read/write
0	m2-3	R
1	cur	R
2	inc	R
3	fin	R
4	PDP	R/W
5	cwd	R
6	in-3	R
7	new	W
10	data-out	W
11	[non utilisé]	
12	mout-3	W
13	phr-1	R
14	fr1-1	R
15	fr2-1	R
16	m1-1	R
17	m2-1	R
20	phw-1	W
21	phr-2	R
22	fr1-2	R
23	fr2-2	R
24	in-1	R
25	phw-2	W
26	wt-1	W
27	m1-2	R
30	m2-2	R
31	mout-1	W
32	[non utilisé]	
33	in-2	R
34	data-in	R
35	wt-2	W
36	m1-3	R
37	mout-2	W

Les cycles "11 et "32, non utilisés, peuvent éventuellement être employés pour des échanges ultérieurs entre unités (cycles DATA-OUT et DATA-IN).

En résumant, on a donc ici en tout, à 16 kHz, 32 unités logiques, 96 multiplicateurs-accumulateurs et 64 additionneurs-soustracteurs à trois entrées avec possibilité de consultation de la Waveform Memory.

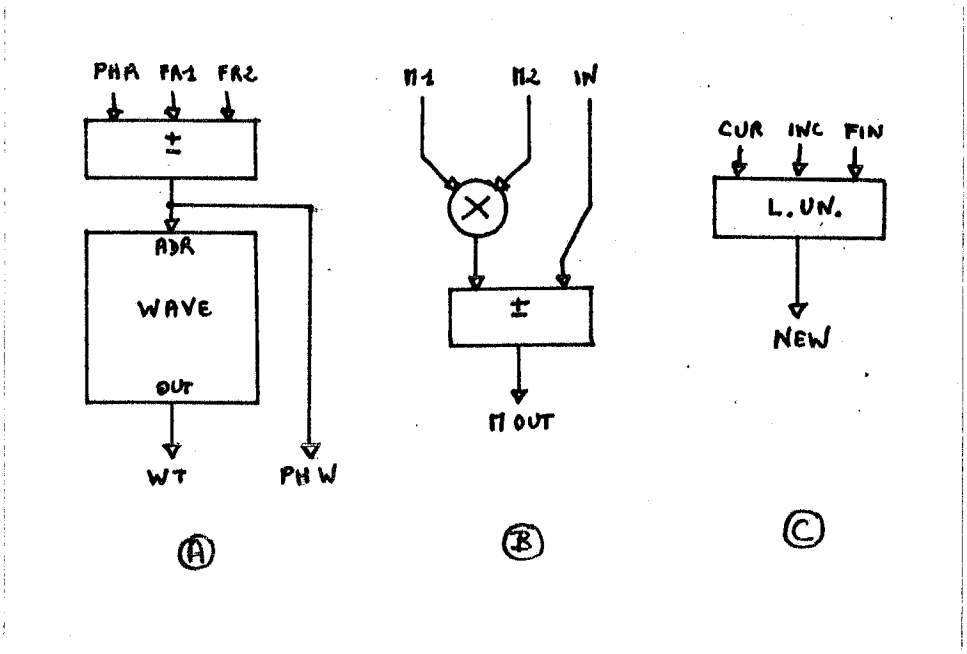

fig. 23 - les modules qu'on peut obtenir d'une 4U avec le microprogramme de la table 3. Dans un cycle de 2 usec on peut engendrer 2 additionneurs-soustracteurs à 3 entrées avec lecture de mémoire, 3 multiplicateurs-accumulateurs et une unité logique. Cette dernière unité est la même qu'en fig. 10

En interconnectant convenablement ces modules de base, on peut obtenir en pratique tous les éléments de synthèse et de traitement du signal qu'on a déjà décrit dans l'Appendice II, à propos de la 4C; comme dernier exemple, dans **[ASTA, V., 1980]** on trouvera une application dans laquelle est réalisé un synthétiseur complet de voix à formants, entièrement programmable.

Pour résumer, une 4U ainsi programmée représente une unité très sophistiquée pour la synthèse avec des techniques (en particulier) soustractives et non linéaires.

Le deuxième microprogramme est montré dans la table 4; il permet de réaliser, en 2 usec, les modules représentés fig. 24 (deux par type).

Ce microprogramme rends la 4U assez proche d'un synthétiseur 4A, puisque comme on le comprend facilement par la figure on peut programmer, à 16 kHz, 64 oscillateurs à double entrée en fréquence (par les unités de type A) et 64 à simple entrée (par les unités de type B); chacun de ces oscillateurs a un multiplicateur d'amplitude et un accumulateur directement câblés

Table 4

numéro de microcycle	opération	read/write
0	frl-a2	R
1	fr2-a2	R
2	amp-b2	R
3	PDP	R/W
4	phw-a2	W
5	[non utilisé]	
6	phr-al	R
7	frl-al	R
10	fr2-al	R
11	in-b2	R
12	phw-al	W
13	amp-a2	R
14	out-b2	W
15	phr-bl	R
16	frl-bl	R
17	in-a2	R
20	phw-bl	W
21	amp-al	R
22	out-a2	W
23	phr-b2	R
24	frl-b2	R
25	in-al	R
26	phw-b2	W
27	cw-bl	R
30	amp-bl	R
31	out-al	W
32	data-out	W
33	in-bl	R
34	data-in	R
35	cw-al,a2,b2	R
36	out-bl	W
37	phr-a2	R

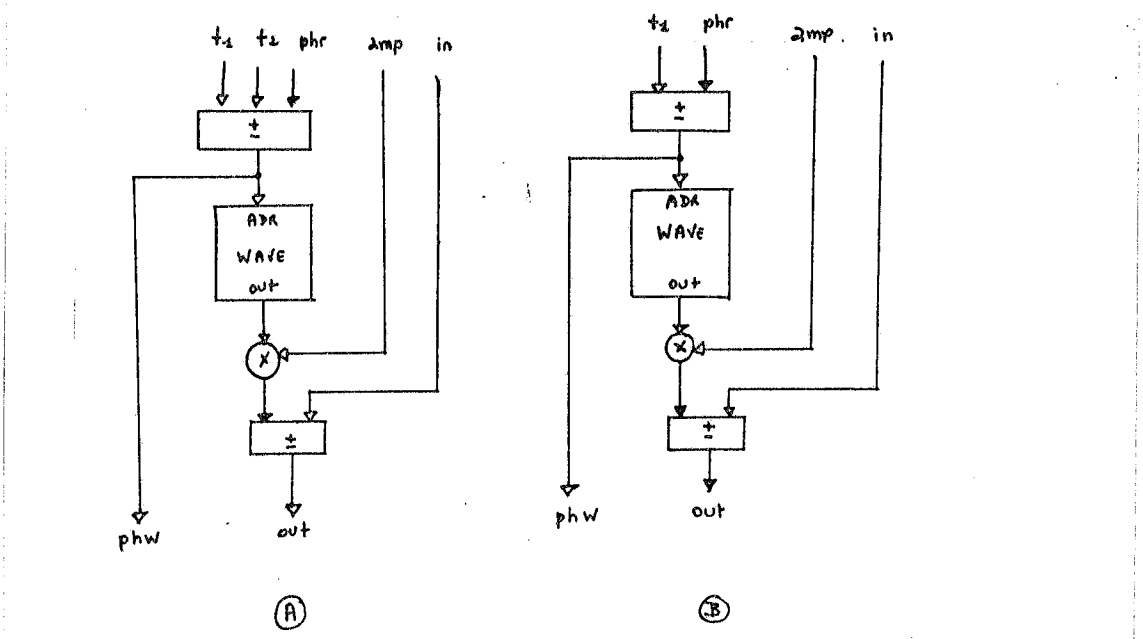

fig. 24 - les modules qu'on peut obtenir d'une 4U avec le microprogramme de la table 4. Dans un cycle de 2 µsec on peut engendrer 2 éléments de chaque type.

dans l'unité, ce qui permet d'économiser des microcycles (qu'on devrait autrement utiliser pour spécifier la connexion, par un registre, entre la sortie de la Wavetable Memory et l'entrée du multiplicateur).

En général, plusieurs oscillateurs utilisent une même forme d'onde; dans cette unité - mais ceci est une caractéristique générale des 4U, indépendante du microprogramme - on peut avoir en même temps jusqu'à 32 formes d'ondes différentes, en divisant la Waveform Memory en blocs de 512 mots chacun.

Pour en revenir aux oscillateurs, en programmer un à simple entrée en fréquence dans cette unité, par exemple, signifie lui associer cinq registres: un pour l'accumulation de la phase, un pour la fréquence, un pour l'amplitude, un pour l'accumulation et un pour la sortie. En même temps, on assigne aussi un registre pour le mot de contrôle de la forme d'onde. On fait cela naturellement avec l'Address Memory. Les registres de fréquence peuvent contenir des valeurs fixes, écrites à l'avance dans la Data Memory, ou bien par exemple des valeurs engendrées par des générateurs d'enveloppes programmés sur d'autres unités 4U. A chaque lecture ou écriture des cinq registres correspond évidemment un microcycle.

Les 4 oscillateurs qu'on peut réaliser dans un cycle de 2 usec requièrent un total de 26 microcycles; les autres cycles sont utilisés pour les opérations d'entrée/sortie sur le bus interne et sur celui du PDP-11, et pour la lecture des mots de contrôle des oscillateurs. En total, cette unité peut réaliser 128 oscillateurs dans un cycle complet, à la fréquence d'échantillonage de 16 kHz.

Ce microprogramme est surtout destiné à la synthèse par des techniques additives, ou par modulation de fréquence (on peut programmer 64 instruments FM complets), ou encore par modulation d'amplitude: par exemple une modulation entre deux oscillateurs est obtenue en spécifiant, comme registre d'amplitude pour le deuxième, celui qui contient la sortie du premier.

Le troisième microprogramme est montré dans la table 5; il est optimisé pour obtenir le maximum possible d'éléments multiplicateur-accumulateur, et rien d'autre: il y en a sept par cycle intermédiaire, donc on peut réaliser 224 éléments de ce type à 16 kHz; ce microprogramme est destiné essentiellement à la réalisation de filtres numériques. Plus précisément, ce microprogramme, plutôt que pour des filtres de type classique, comme on peut les trouver par exemple en **[RABINER, L., GOLD, R., 1975]**, a été conçu pour un nouveau type de filtres, récemment mis

au point à l'IRCAM, et brièvement présenté en **[ASTA, V., 1980]**. Il s'agit essentiellement d'une transposition numérique des "filtres universels" analogiques bien connus, synthétisés par variable d'état, qui ont déjà trouvé d'intéressantes applications musicales dans les synthétiseurs analogiques **[COLIN, D.P., 1971; ORR, T., THOMAS, D.W., 1973]**, et qui sont obtenus grâce à une transformée z particulière. Ces filtres se sont montrés en pratique extrêmement efficaces et polyvalents, au moins pour les applications en cause. Avec quatre éléments de calcul de ce microprogramme on peut réaliser une cellule du 2.ème ordre (avec des zéros à l'infini), munie de gain d'entrée et avec sorties passe-bas, passe-haut et passe-bande disponibles simultanément. Dans une carte on pourra donc avoir jusqu'à 54 cellules du 2.ème ordre à 16 kHz. Ce microprogramme est donc particulièrement adapté pour la synthèse avec des techniques de type soustractif **[CANN, R., 1979; MOORER, J.A., 1977]**.

Table 5

numéro de microcycle	opération	read/write
0	m2-6	W
1	in-6	R
2	out-5	R
3	PDP	R/W
4	m1-7	R
5	m2-7	R
6	in-7	R
7	out-6	W
10	m1-1	R
11	m2-1	R
12	in-1	R
13	out-7	W
14	m1-2	R
15	m2-2	R
16	in-2	R
17	out-1	W
20	[non utilisé]	
21	m1-3	R
22	m2-3	R
23	in-3	W
24	out-2	R
25	m1-4	R
26	m2-4	R
27	in-4	R
30	out-3	W
31	m1-5	W
32	data-out	W
33	m2-5	W
34	data-in	R
35	in-5	R
36	out-4	W
37	m1-6	W

Passons maintenant à l'examen du dernier microprogramme, montré dans la table 6 et permettant de réaliser les modules de la fig. 25; dans un cycle de 2 usec, on dispose d'un seul module pour chaque type; ceux marqués avec les lettres B et D, et l'unité logique, étaient déjà présents dans certains des microprogrammes illustrés précédemment, tandis que les autres sont de type nouveau. Cette configuration de la 4U permet essentiellement de réaliser toutes les unités déjà vues (en plus petite quantité), mais donne la possibilité de faire beaucoup d'autres choses.

Table 6

numéro de microcycle	opération	read/write
0	wt-b	W
1	cur	R
2	inc	R
3	fin	R
4	PDP	R/W
5	m1-d	R
6	m2-d	R
7	new	W
10	cwd	R
11	in-d	R
12	phr-a	R
13	mout-d	W
14	fr1-a	R
15	fr2-a	R
16	phr-b	R
17	phw-a	W
20	fr1-b	R
21	fr2-b	R
22	m11-e	R
23	phw-b	W
24	m12-e	R
25	phr-c	R
26	m2-e	R
27	fr-c	R
30	wt-a	W
31	phw-c	W
32	wtin-c	W
33	data-out	W
34	in-e	R
35	data-in	R
36	shr-a	W
37	mout-e	W

Les éléments originaux sont ceux marqués avec A et C: le premier permet d'utiliser le registre à décalage. Le deuxième permet d'écrire, aux adresses désirées, dans la Waveform Memory: il est ainsi possible d'utiliser cette mémoire comme élément de retard de durée variable (jusqu'à une seconde à 16 kHz), chose très utile en particulier pour la réalisation d'unités de réverbération artificielle. Les travaux de SCHROEDER **[M.R., 1962]**

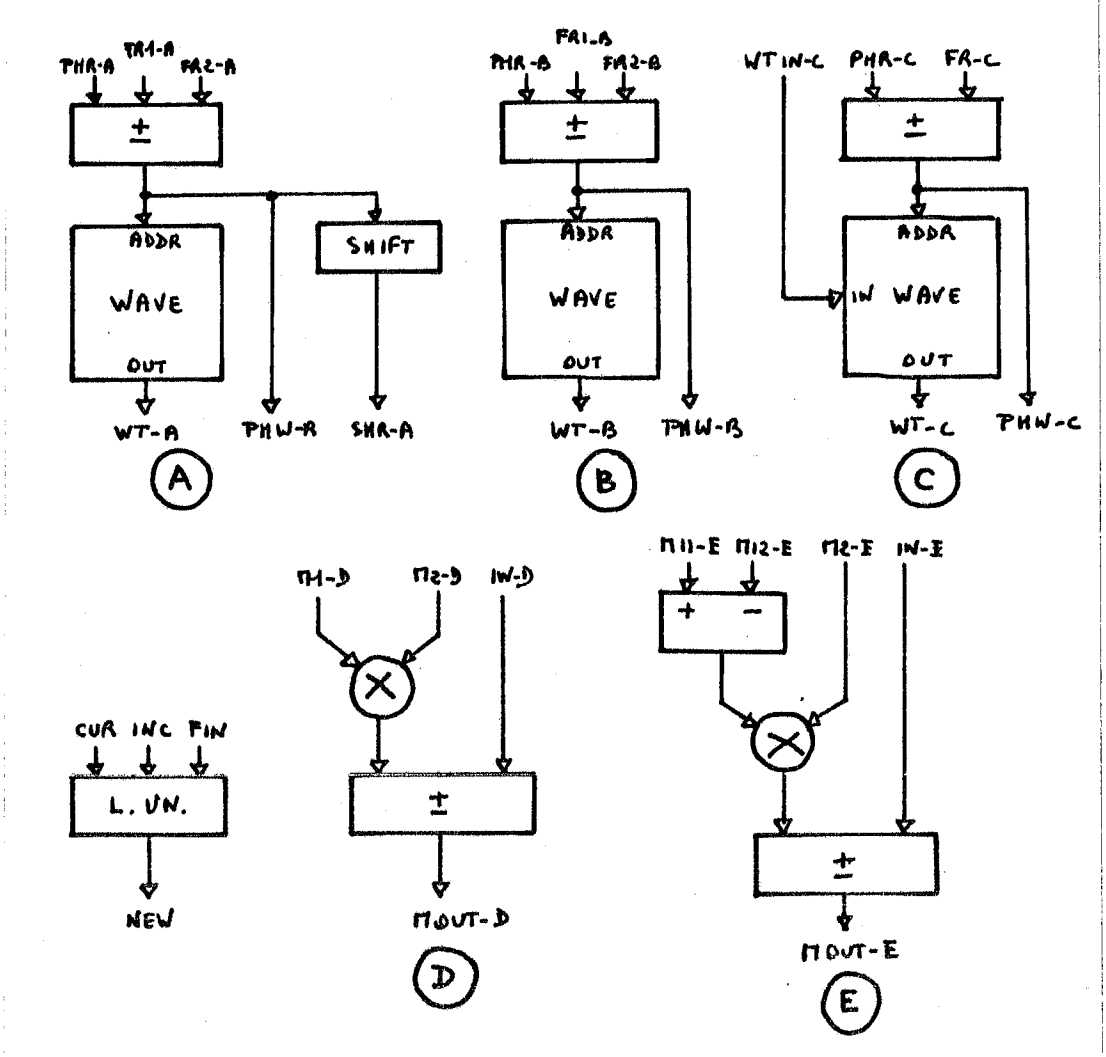

fig. 25 - les modules qu'on peut obtenir d'une 4U avec le microprogramme de la table 6. Dans un cycle de 2 usec on peut engendrer un élément de chaque type montré. L'unité logique est la même qu'en fig. 10

et de MOORER **[J.A., 1977 et 1979]** ont montré qu'on peut obtenir digitalement des unités de reverbération artificielle avec des

combinaisons appropriées de filtres passe-tout et de filtres à peigne, utilisant, tous les deux, des éléments de retard de l'ordre des dizaines de millisecondes.

Si on utilise une unité de type A, une de type B, et une de type E, on peut réaliser un oscillateur à interpolation (on peut donc en avoir 32 à 16 kHz), dont la sortie est calculée par interpolation linéaire de deux échantillons contigus de la forme d'onde, sur la base de la valeur fractionnaire de la phase actuelle. Ceci donne la possibilité d'utiliser des blocs de forme d'onde très petits (par exemple 512 mots), car le rapport signal-bruit est plus favorable **[MOORE, F.R., 1977].**

Le microprogramme de la table 6 permet aussi de mettre en oeuvre un algorithme de transformée de Fourier rapide (FFT) à entrelacement fréquentiel **[RABINER, L., GOLD, R., 1975]:** en un cycle complet on peut effectuer le calcul de deux butterflys (y compris les lectures de tableaux et autres opérations nécessaires pour les fenêtres temporelles, les exponentielles complexes, le "bit reversal", et les logarithmes et racines carrées nécessaires pour le calcul du module); on peut traiter, à 16 kHz de fréquence d'échantillonnage, 256 points complexes, pour une résolution en fréquence de 63 Hz. Les fenêtres de signal d'entrée peuvent être mises à jour toutes les 32 msec. Les deux butterflys requièrent environ un quart des ressources disponibles sur une carte 4U.

Si on utilise un microprogramme spécialisé, à 1024 microcycles, et une strucure en pipeline de l'algorithme, on peut arriver à calculer 10 butterflys en 62 usec, donc on peut avoir une FFT à 1024 points complexes entièrement en temps réel (les fenêtres de signal sont mises à jour à chaque échantillon).

En général, avec l'utilisation de microprogrammes ad hoc, on peut mettre en oeuvre toutes sortes d'algorithmes d'analyse ou de traitement du signal: avec une carte 4U on peut effectuer une des opérations suivantes **[FAVREAU, E., 1982]:**

- auto-correlation ou correlation croisée, jusqu'à 62 coefficients;
- analyse spectrale par FFT (déjà mentionnée);
- analyse spectrale par prédiction linéaire (LPC) **[MARKEL, J.D., GRAY,A.H., 1976]**, jusqu'à l'ordre 25;
- détecteur de fondamental;
- transformée de Hilbert;
- filtres de Butterworth, Tchebicheff, elliptiques, jusqu'à l'ordre 32 [26];
- synthèse de la parole basée sur l'analyse LPC, par des filtres en treillis jusqu'à l'ordre 32.

III.6 La version industrielle

Comme on l'a déjà anticipé dans l'introduction à cet Appendice III et dans la note [21], après la mise au point du prototype, une version industrielle de la 4X a été réalisée à la suite d'un accord entre l'IRCAM et la société SOGITEC. Cette version, qui est maintenant dans le commerce, présente un certain nombre d'améliorations et de variantes par rapport au prototype, qu'on décrira synthétiquement ici.

Le changement le plus important concerne l'interfaçage du système à l'ordinateur hôte: la Bus Terminator Unit est remplacée par une carte d'interface avec un bus plus moderne et flexible que l'UNIBUS: le VME-bus **[GIROD, D., 1982]**. Une configuration type pour la 4X prévoit un processeur Motorola MC68000 pour le contrôle en temps réel du système, plus un deuxième MC68000, avec un système d'exploitation multi-tâches et éventuellement multi-utilisateur, utilisable comme station de dévelopement du software pour le système.

Le bus VME possède 32 lignes d'adressage, contre les 18 de l'UNIBUS, et peut donc adresser jusqu'à 4 Giga octets (contre 256 k octets). Ceci a comporté, entre autre, la disparition de l'unité de Memory Management (dans l'Interface Unit), étant donné que, avec le grand espace d'adressage dont on dispose, on peut se permettre d'avoir toutes les mémoires de la 4X directement sous le contrôle de la CPU, sans avoir recours à une fenêtre commune d'adressage.

Aussi, grâce à une meilleure intégration des composants sur les cartes, et à la simplification de toute la logique de commande dûe à l'absence du Memory Management, il a été possible de concentrer les deux cartes de contrôle (Interface Unit et Interconnection Unit) en une seule.

Finalement, l'apparition sur le marché de composants électroniques toujours plus flexibles et rapides, et de mémoires à très grande capacité, a permis d'introduire les extensions suivantes:

- la taille de la Data Memory a été portée à 1024 mots de 24 bits en standard, extensibles à 4096 mots (contre 512 pour le prototype);

- par conséquent, la largeur de mot de l'Address Memory a été étendue à 12 bits;

- les Waveform Memories ont été portées à une taille de 64 k octets (contre 16 k octets dans le prototype); ceci implique que le registre R1 qui les adresse (voir fig. 21) est maintenant à 16 bits (14 dans le prototype, pour des mémoires de 16 k octets). Avec de telles mémoires, on peut réaliser des unités de retard de durée jusqu'à 4 secondes (à 16 kHz; voir paragraphe "microprogrammes et applications").

- de même, le buffer de l'Interface Unit a été porté aussi à 64 k octets; analoguement, les registres d'index qui le gèrent (R3, R4 et R5; voir fig. 21) sont maintenant à 16 bits (14 dans le prototype);

- un nouveau registre, dénommé STORE REG, a été introduit sur le C-BUS des unités de synthèse 4U (voir fig. 18): ce registre est simplement une unité de stockage temporaire, permettant de tenir à disposition sur le C-BUS une donnée utilisée plusieurs fois dans un certain microprogramme, sans devoir la faire transiter à chaque fois par la Data Memory, M-OUT etc.; ceci permet d'optimiser ultérieurement certains microprogrammes;

- les timers du système peuvent être soit incrémentés, soit décrémentés, sous le contrôle du microprogramme;

- le nombre de canaux d'entrée-sortie du système a été augmenté, avec une plus grande flexibilité d'usage: dans le prototype on peut avoir 16 DACs et 16 ADCs, dans la version industrielle on peut choisir n'importe quelle combinaison de DACs et ADCs jusqu'à un total maximum de 64 canaux;

- une nouvelle micro-instruction a été introduite dans le système, permettant d'exécuter un saut inconditionnel à zéro dans l'adressage de l'Address Memory. Ceci amène qu'un cycle complet des cartes de synthèse n'est plus nécessairement composé de 1024 microcycles de base, mais d'un nombre arbitraire de ceux-ci (entre 1 et 1024). La conséquence, très importante, de cette micro-instruction est que la fréquence d'échantillonnage du système n'est plus nécessairement un multiple de 16 kHz, mais peut varier beaucoup plus finement en fonction des exigeances du compositeur.

Notes

[1] précisons que le modulateur en anneau est essentiellement un multiplicateur analogique à 4 cadrans, où un des deux signaux d'entrée doit être forcément sinusoïdal; le VCA est normalement un multiplicateur analogique à deux cadrans (plus rarement 4), c'est à dire où une des deux entrées doit être toujours positive.

[2] les seules techniques qu'on peut implémenter sur un synthétiseur classique sont la synthèse soustractive (un VCF qui filtre dynamiquement un VCO) et la synthèse non-linéaire par modulation d'amplitude (deux VCO qui vont à un modulateur en anneau). La synthèse par modulation de fréquence, bien que théoriquement possible, fournit rarement des bons résultats à cause des imprécisions des fréquences des oscillateurs analogiques qui ne permettent jamais d'avoir une superposition exacte des composantes repliées autour de 0 Hz.

[3] rappelons que tous les dispositifs électroniques numériques - et en particulier les ordinateurs - utilisent des représentations des nombres en base 2, pour limiter les chiffres à 0 et 1, qu'on peut associer à deux différents niveaux de tension électrique.

[4] cela est vrai, naturellement, si on considère la génération d'une seule voix: il est évident que plusieurs voix demandent un travail additionnel d'enregistrement et de mixage qui n'est pas en temps réel.

[5] les dispositifs de contrôle gestuel utilisés avec des synthétiseurs numériques sont, dans la plus part des cas, des systèmes analogiques générant une tension de contrôle qui est convertie en forme numérique à une cadence adéquate, normalement assez inférieure à celle des signaux acoustiques de sortie (étant donné que les signaux de contrôle varient normalement beaucoup plus lentement que les signaux audio).

[6] on pense évidemment aux ordinateurs de gamme moyenne (mini-ordinateurs), conçus pour une utilisation mono-utilisateur. En effet, les échantillons du signal devant sortir à une cadence bien précise, il n'est pas possible de réaliser un instrument en temps réel, si simple soit-il, avec un ordinateur multi-tâche, devant servir plusieurs programmes à la fois et donc ne pouvant pas garantir sa disponibilité à tous les instants de sortie d'un échantillon.

[7] le tronquage n'est pas la seule méthode adoptée pour obtenir l'adresse effective de l'échantillon à partir de la phase: on utilise aussi l'arrondi de la phase à l'entier le plus proche, ou ancore on interpole linéairement entre les valeurs des deux échantillons qui précedent et qui suivent la valeur de la phase. Dans la deuxième technique, la table

l se modifierait comme on le montre dans la table 7 (un * marque les valeurs qui ont changé).

Table 7

numéro d'échantillon	phase instantanée	adresse de l'échantillon
1	6.4	6
2	12.8	13 *
3	19.2	19
4	25.6	26 *
5	32.0	32
6	38.4	38
7	44.8	45 *

Le programme montré en Appendice I présenterait la modification d'une ligne: au lieu de l'instruction

```
      IADR=IFIX(PHASE)
```

on aurait

```
      IADR=IFIX(PHASE + 0.5)
```

Dans la technique par interpolation, ce même programme présenterait quelques complications: au lieu des deux instructions

```
      IADR=IFIX(PHASE)
      IOUT=IWAVE(IADR)
```

on aurait

```
      IADR=IFIX(PHASE)
C contrôle pour interpolation entre le dernier
C et le premier echantillon
      IF (IADR-LENWAV) 1,2,2
1     IADR1=IADR+1
      GO TO 3
2     IADR1=1
C calcul de la pente pour l'approximation linéaire entre les
C deux echantillons.
3     SLOPE=FLOAT(IWAVE(IADR1)-IWAVE(IADR))
C détermination de la partie fractionnaire de la phase
      FRACT=PHASE-FLOAT(IADR)
C détermination de la variation de l'echantillon due
C à l'interpolation
      DELTA=SLOPE*FRACT
C calcul de l'echantillon de sortie
      IOUT=IWAVE(IADR)+DELTA
```

Les trois techniques demandent progressivement plus de temps de calcul, mais permettent d'utiliser moins de mémoire pour le stockage des formes d'onde, à égalité de résultats. A ce propos, MOORE **[F.R., 1977]** a introduit des tables

expérimentales sur les rapports signal/bruit obtenus avec les trois techniques, et a montré l'existence de relations entre la longueur des tableaux et la précision optimale (en nombre de bits) des échantillons mémorisés.

[8] nous ne parlerons pas en détail de ces programmes, qui n'intéressent pas la synthèse en temps réel; il suffira ici de rappeler les plus importants: Music V (le premier programme universellement connu, et le plus répandu encore aujourd'hui) **[MATHEWS,M.V., et al., 1969]**, Music IV-BF **[HOWE, H., 1975]**, Music 360 **[VERCOE,B., 1971]**, Music 11 **[VERCOE, B., 1980]**, Music 10 **[BATTIER, M., 1980]**.

[9] les deux premiers types de synthèse sont souvent indiqués comme synthèses linéaires, alors que les deux dernières sont aussi indiquées sous le nom de synthèses non linéaires; cela est fait avec référence au type de transformations qu'on opère sur les sons élémentaires de base pour obtenir le résultat voulu.

Il existe naturellement d'autres types de synthèse, outre celles dont on parlera ici; en particulier, une technique extrêmement simple et courante est la synthèse (non linéaire) par modulation d'amplitude, obtenue par multiplication des sorties de deux oscillateurs entre elles; par contre, des techniques relativement très complexes, basées sur certains développements en série de fonctions trigonométriques - tout comme la technique de modulation en fréquence - ont été introduites par MOORER **[J.A., 1976]**; on rappellera également, entre les techniques plus récemment proposées et destinées probablement à être très utilisées dans le futur, celle développée par JUSTICE **[J.H., 1979]**, basée sur la transformée de Hilbert discrète, et celle introduite par ROZENBERG **[M., 1980]**, basée sur des techniques de balayage linéaire en fréquence.

[10] un exposé plus exhaustif de cet argument peut être trouvé dans **[MOORER, J.A., 1977]**.

[11] on sait **[RABINER, L., GOLD, R., 1975]** qu'en général un filtre à p poles et q zéros, réalisé sous forme numérique, peut être représenté, dans le domaine du temps, par une relation du type de la (5) entre sa sortie X(n) et son entrée U(n), où les coefficients a_k déterminent l'emplacement des poles, et les coefficients b_r l'emplacement des zéros du filtre. Les a_k et b_r sont respectivement les coefficients du dénominateur et du numérateur de la fonction de transfert du filtre exprimée dans le domaine de la variable complexe z. Dans la (5) ces coefficients sont fonctions du temps n, puisque le filtre est variable.

[12] dans le cas le plus simple, U(n) sera la sortie d'un oscillateur à forme d'onde complexe:

$$(5') \qquad U(n)=wav\ (\omega(n)\ nT + \varphi\ )$$

avec $\omega(n) = 2\pi f(n)$, f(n) fréquence de l'oscillateur et donc de tout l'instrument, fonction (lente) du temps; en général, naturellement, U(n) peut être le résultat partiel d'un ensemble de modules interconnectés entre eux.

[13] précisons que dans la voix humaine le système d'excitation n'est pas toujours constitué par les cordes vocales: par exemple, dans la production des consonnes fricatives non voisées (f, s, ch) la source sonore consiste en un bruit d'écoulement de l'air au passage par une constriction du conduit vocal. Pour une introduction détaillée au mécanisme de la phonation humaine, voir **[CHAPMAN, W.,1971; LIENARD, J.S., 1977]**.

[14] plus précisément, on peut montrer **[PERONI, B., 1973]** que entre l'index de modulation I et le coefficient de conversion amplitude-fréquence A_m subsiste la relation

$$(6') \qquad I = A_m / f_m$$

(Cette formule n'est valide, à la rigueur, qu'en cas de modulante sinusoïdale: voir note [16]). Puisque tous les raisonnements sur le spectre engendré se font en fonction de l'index de modulation, l'instrument de fig. 1 requiert que l'ordinateur qui contrôle les paramètres de l'instrument effectue de lui-même la multiplication I x f_m; autrement, si on dispose d'un autre multiplicateur dans le synthétiseur, on peut utiliser l'instrument de fig. 26, beaucoup plus commode en pratique car il permet de raisonner exclusivement en termes d'index de modulation. Il faut ajouter que les deux enveloppes des fréquences se réduisent souvent à une seule, car ces deux paramètres sont normalement soit identiques, soit en un rapport fixe entre eux.

[15] plus précisément, toujours dans l'hypothèse de pouvoir négliger les contributions du spectre dépendant des fonctions de Bessel d'ordre supérieur à I+1, la largeur de bande totale du signal peut être calculée par la formule approchée:

$$(7') \qquad BW = 2\ f_m\ (\ I + 1\ )$$

valable pour f_m $I < f_c$, $I > 1$ (ce qui implique, entre autre, $f_m < f_c$; la bande latérale inférieure du spectre n'arrive jamais à 0 Hz, et donc il n'y a pas de repliement du spectre).

La (7') est un cas particulier d'une relation plus générale, connnue sous le nom de "règle de Carson" **[PERONI, B., 1973]**; pour les applications musicales, on peut donner une formule, beaucoup plus utile en pratique, pour le contenu spectral maximum du signal (plus haute fréquence non

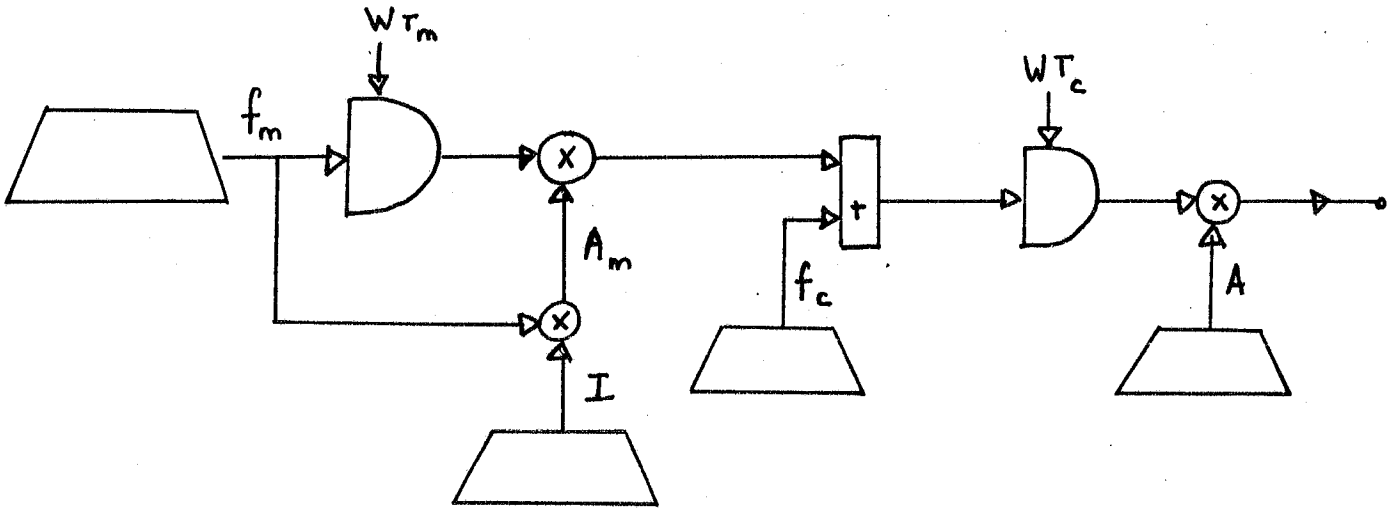

fig. 26 - instrument de base pour synthèse FM, commandé directement par l'index de modulation

négligeable du signal):

$$(7'') \qquad Fmax = f_c + f_m\,(\,I + 1\,) = (\,f_c + f_m\,) + f_m\,I$$

Cette dernière formule est valable pour toutes les valeurs de f_m et f_c, et donc même en cas de repliement du spectre; elle montre entre autres que la relation entre Fmax et I est pratiquement linéaire.

[16] l'utilisation d'une forme d'onde non sinusoïdale pour l'oscillateur modulant complique considérablement le traitement mathématique de cette technique; en effet (on fait le raisonnement dans le domaine du temps continu, pour simplicité d'exposition), étant donné que la fréquence (plus exactement, la pulsation) est la dérivée dans le temps de la phase instantanée, l'expression générale d'un signal modulé en fréquence est la suivante (en assumant une dépendance linéaire de la pulsation par rapport à l'oscillateur modulant) **[PERONI, B., 1973]**

$$(9') \qquad X(t) = A\ car\left[\,\omega_c\,t + K\int_{t_0}^{t} mod(t)\,dt + \varphi_c\,\right]$$

où car(.) et mod(.) sont les deux fonctions (a priori quelconques) respectivement de porteuse et de modulante. Si mod(.) est sinusoïdale, son intégrale l'est aussi, et on peut arriver facilement à l'expression beaucoup plus simple

(9") $X(t) = A \cos [\omega_c t + I \sin (\omega_m t + \varphi_m) + \varphi_c]$

où I est donné par la (6') (avec $A_m = K / 2\pi$); la (9") se réduit évidemment à la (6), dans le cas de porteuse sinusoïdale (en passant au temps discret et en négligeant les phases initiales des oscillateurs).

De toute façon, en décomposant la forme d'onde modulante en une somme de sinusoïdes, on peut encore dériver une formule pour le spectre total obtenu **[LE BRUN, M., 1977]**.

[17] l'IRCAM (Institut de Recherche et Coordination Acoustique/Musique), dirigé par Pierre Boulez, est le plus gros centre de recherche en synthèse musicale existant en Europe, et un des plus importants dans le monde. Tous les synthétiseurs qui y ont été réalisés sont essentiellement l'oeuvre de Giuseppe Di Giugno, ancien professeur de physique à l'université de Naples (Italie), qui a conçu les systèmes et a effectué le projet de toutes les unités; plusieurs chercheurs - dont l'auteur de ces lignes - ont été chargés d'écrire des programmes de contrôle de ces machines, pour plusieurs applications, sous la coordination de Jean Kott.

[18] pour une explication détaillée de cet effet, typique de tout système numérique, voir **[RABINER, L., GOLD, R., 1975]**, chap. 2, pag. 9 - 74.

[19] le bus d'un ordinateur (ou de n'importe quel système électronique numérique) est l'ensemble de signaux (données, adresses et signaux de contrôle) nécessaires pour le transfert des données entre les différents blocs du système.

[20] cela ne veut pas dire, naturellement, que tout utilisateur doit écrire un programme à chaque fois qu'il a besoin, par exemple, d'un oscillateur: il existe, à ce propos, plusieurs programmes de contrôle de la 4C, développés à l'IRCAM - certains sont d'usage général, d'autres sont limités à des applications particulières -, conçus dans le but de simplifier l'utilisation du système et d'en accentuer la versatilité **[CAPPIELLO, C., 1980; ASTA,V., 1979 et 1980; PREVOT, Ph., 1980 et 1981; ABBOTT, C.W., 1980]**. Au niveau des interconnexions, par exemple, ces programmes prévoient des commandes spécifiques pour préparer directement les unités les plus courantes; une syntaxe possible d'une commande pour l'interconnexion d'un oscillateur est la suivante:

OSC<n> <out>,<wt>,<frq1>[,<frq2>,<amp>,<add>

où:

- <n> est le numéro d'unité additionneur-lecteur de forme d'onde à utiliser pour construire l'oscillateur;

- <out> est le numéro (ou le mnémonique) du registre qui doit mémoriser la sortie de l'oscillateur (voir Appendice II pour plus de détails);

- <wt> est le numéro de la forme d'onde à utiliser;

- <frq1> est le numéro du registre qui contient la fréquence de l'oscillateur (ou directement sa valeur); ce registre peut être la sortie d'une unité logique utilisée comme générateur d'enveloppe;

- <frq2> est le numéro du registre ou la valeur pour la deuxième fréquence (par exemple une fréquence de modulation de l'oscillateur);

- <amp> est le numéro du registre ou la valeur pour l'amplitude de l'oscillateur;

- <add> est le numéro du registre ou la valeur qui doit être accumulée avec la sortie de l'oscillateur;

- tous les arguments après [sont optionnels.

Si <amp> et/ou <add> sont spécifiés, le programme utilisera automatiquement une unité multiplicateur-additionneur et l'interconnectera à l'oscillateur, pour lui fournir un contrôle de l'amplitude.

Il y aura aussi, naturellement, une syntaxe adéquate pour la spécification des listes de paramètres (partition).

[21] les descriptions de la 4X fournies dans cet exposé (dans ce paragraphe et dans l'Appendice III) se réfèrent à la mise en oeuvre du premier prototype, connu aussi sous le nom de 4X/001 et réalisé entièrement à l'IRCAM; la société SOGITEC a conclu en suite un accord avec l'IRCAM même, pour la réalisation de la version industrielle du système 4X; cette version (4X/002) présente un certain nombre de variantes et améliorations par rapport au prototype (voir Appendice III, dernier paragraphe).

[22] il n'est pas difficile de se convaincre que l'opération exécutée par l'unité logique permet, en connectant la sortie <curw> et l'entrée <curr> à un même registre, de réaliser des rampes, à partir de la valeur initiale de <curr> jusqu'à la valeur de <fin>, à une vitesse qui dépend de la valeur d'incrément déposée en <inc> (voir fig. 27). Une fois atteinte la valeur finale, la rampe reste bloquée à cette valeur, jusqu'à ce que l'ordinateur maître chargera des nouvelles données d'incrément et de valeur finale, pour démarrer une nouvelle rampe. L'instant de changement de ces

valeurs peut être fixé à l'avance en chargeant un timer, qui provoquera une interruption après le temps voulu. Dans la routine d'interruption, l'ordinateur maître rechargera les valeurs pour la nouvelle rampe et pour la prochaine interruption du timer. Par cette technique, on peut engendrer des enveloppes de forme quelconque, approximées par segments.

[23] là aussi vaut l'observation faite à propos de la 4A et de la 4C, que tout ordinateur PDP-11 peut être connecté sans aucune adaptation au système. La version industrielle de la 4X (voir Appendice III, dernier paragraphe) prévoit une interface avec un processeur Motorola 68000, par le bus VME; cet ordinateur présente l'avantage, par rapport aux PDP-11, de pouvoir adresser un espace-mémoire beaucoup plus important, ce qui permet d'éviter le multiplexage des mémoires du système, qui peuvent être toutes "mappées" en même temps dans la mémoire centrale du 68000. Le software de contrôle qui en résulte est donc assez simplifié.

[24] les définitions de microcycle de base, cycle intermédiaire et cycle complet ont été données dans l'Appendice II à propos de la 4C, et restent valides pour les 4U; on souligne cependant que le cycle intermédiaire des 4U, en tant que temps d'adressage de toute la Program Memory, peut avoir en pratique des durées différentes, étant donné que la Program Memory, comme on le détaillera par la suite, existe en plusieurs options permettant de lui attribuer des différentes longueurs.

[25] la réalisation de cette carte est strictement dépendante du bus de l'ordinateur avec lequel la 4X est à interfacer; la carte existe donc, à l'heure actuelle, en deux différentes versions, une pour l'UNIBUS (prototype), et l'autre pour le VME bus (version industrielle). Voir note [23] et Appendice III, dernier paragraphe.

[26] la puissance de calcul d'une carte 4U permet, en effet, l'implémentation de filtres numériques, de type classique ou en treillis, d'ordre jusqu'à 32; ceci dit, pour des ordres aussi élevés on rencontre des limitation pratiques, dûes essentiellement à la longueur de mot des coefficients des multiplicateurs (16 bits). Ceci introduit un bruit (round-off noise) **[RABINER, L., GOLD, R., 1975]** qui produit une dégradation de la qualité du signal filtré.

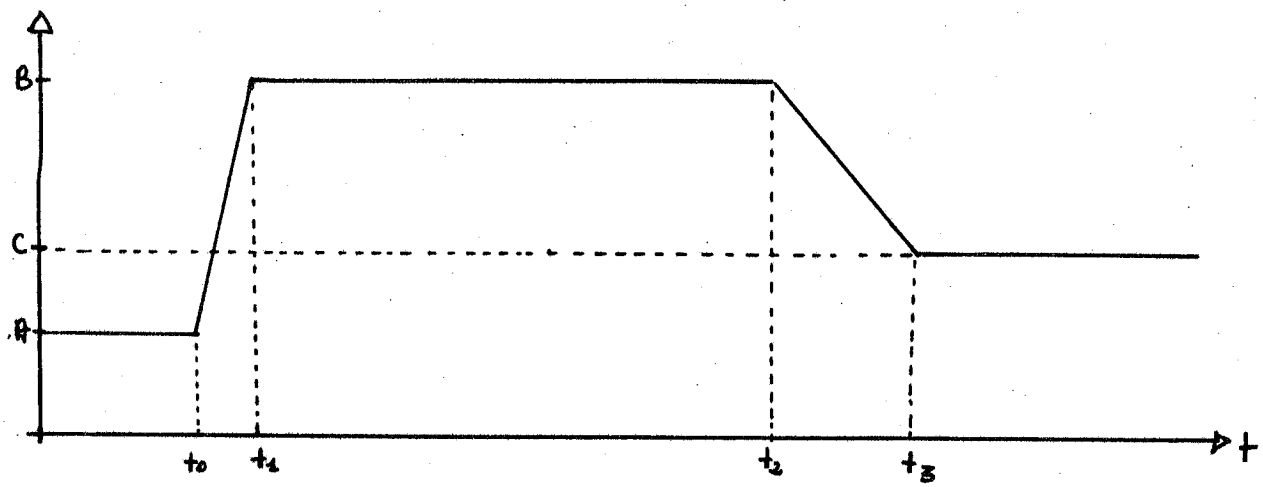

fig. 27 - génération des enveloppes par segments. Au temps t_0, dans l'unité logique, <cur> contient la valeur A; l'ordinateur dépose la valeur B en <fin>, et une valeur (positive) adéquate en <inc>, qui détermine la pente de la rampe et donc l'instant auquel elle arrive à la valeur finale B: <inc> est évidemment donnée par la relation <inc> = (B - A) / (t_1 - t_0).
L'ordinateur initialise aussi un timer à une valeur de temps égale à t_2 - t_0. Au temps t_1, à cause de la valeur en <inc>, la rampe arrive à la valeur B, après quoi elle y reste. Au temps t_2 le timer génère une interruption, et l'ordinateur initialise un nouveau segment de la rampe, en déposant la valeur C en <fin>, et une valeur (négative) en <inc> donnée cette fois par la <inc> = (C - B) / (t_3 - t_2). De même, un timer sera initialisé à la valeur nécessaire pour avoir une nouvelle interruption au moment où il faudra démarrer un nouveau segment; ce temps t_4, normalement, sera supérieur ou égal à t_3.

Bibliographie

- ABBOTT, C.W., 1980, "The 4CED program", proceedings of the 1980 International Computer Music Conference (ICMC), Flushing, New York

- ANGOT, A., 1972, "Compléments de mathématiques", Masson, Paris

- ARFIB, D., 1979, "Digital synthesis of complex spectra by means of multiplication of non-linear distorted sine waves", Journal of Audio Engineering Society (JAES), vol. 27, n. 10

- ASTA, V., 1978, "Tecniche di sintesi per la musica elettronica: la situazione attuale e i recenti sviluppi all'IRCAM", Atti del 6. Convegno dell'Associazione Italiana di Acustica (AIA), Ivrea

- ASTA, V., 1979, "Les systèmes d'interconnexion pour la 4C actuellement existant à l'IRCAM", rapport interne IRCAM

- ASTA, V., 1980, "Unità elettronica per la modifica in tempo reale di suoni strumentali in segnale vocale", Atti del 7. Convegno AIA, Siena

- ASTA, V., CHAUVEAU, A., DI GIUGNO, G., KOTT, J., 1980, "Il sistema di sintesi digitale in tempo reale 4X", Automazione e strumentazione, vol. 28, n. 2

- BATTIER, M., 1980, "Le langage de synthèse Music 10", IRCAM, Paris

- CANN, R., 1979, "An analysis/synthesis tutorial", Computer Music Journal (CMJ), vol. 3, nn. 3-4 et vol. 4, n. 1

- CAPPIELLO, C., 1980, "Un programma interattivo per la gestione di un minicalcolatore veloce", thèse de doctorat en physique, Université de Naples

- CARRE, R., HATON, J.P., LIENARD, J.S., 1979, "Reconnaissance et synthèse de la parole, état de la recherche et du développement", Les synthèses du SESORI, Paris

- CHADABE, J., 1972, "Le principe du 'voltage control' et ses implications pour le compositeur", Musique en jeu, n. 8, pag. 36 - 40, Seuil, Paris

- CHAPMAN, W., 1971, "Introduction to practical phonetics", Summer Institute of Linguistics, Merstham

- CHOWNING, J.M., 1973, "The synthesis of complex audio spectra by means of frequency modulation", JAES, vol. 21, n. 7

- COLIN, D.P., 1971, "Electrical design and musical applications of an unconditionally stable combination voltage controlled filter - resonator", JAES, vol. 19, n. 11

- DI GIUGNO, G., 1976, "A 256 digital oscillators bank", proceedings of the 1976 ICMC, Massachusetts Institute of Technology, Cambridge

Bibliographie

- DI GIUGNO, G., KOTT, J., 1980 a, "L'unité statique 4X", rapport interne IRCAM

- DI GIUGNO, G., KOTT, J., 1980 b, "The IRCAM real-time digital sound processor", proceedings of the 1980 ICMC, Flushing, New York

- DI GIUGNO, G., KOTT, J., 1981, "Présentation du système 4X, processeur numérique de signal en temps réel", rapport IRCAM n. 32/81

- FAVREAU, E., 1982, "Digital signal processing on the real-time processor 4X", proceedings of the 1982 ICMC, La Biennale, Venice

- GIROD, D., 1982, "Le bus VME: description de la partie matérielle et principales caractéristiques", Minis et Micros, n. 160

- HOWE, H., 1975, "Electronic music synthesis", Norton & C., New York

- IMUG, 1983, "MIDI 1.0 specification", Los Altos, California

- JUSTICE, J.H., 1979, "Analytic signal processing in music computation", IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 27, n. 6

- LE BRUN, M., 1977, "A derivation of the spectrum of FM with a complex modulating wave", CMJ, vol. 1, n. 4

- LE BRUN, M., 1979, "Digital waveshaping synthesis", JAES, vol. 27, n. 4

- LIENARD, J.S., 1977, "Le processus de la communication parlée", Masson, Paris

- MARKEL, J.D., GRAY, A.H., 1976, "Linear prediction of speech", Springer-Verlag, Berlin

- MATHEWS, M.V., MILLER, J.E., MOORE, F.R., PIERCE, J.R., RISSET, J.C., 1969, "The technology of computer music", MIT press, Cambridge

- MOOG, R.A., 1965, "Voltage-controlled electronic music modules", JAES, vol. 13, n. 3

- MOORE, F.R., 1977, "Table lookup noise for sinusoidal digital oscillators", CMJ, vol. 1, n. 2

- MOORER, J.A., 1976, "The synthesis of complex audio spectra by means of discrete summation formulas", JAES, vol. 24, n. 11

- MOORER, J.A., 1977, "Signal processing aspects of computer music: a survey", proceedings of the IEEE, vol. 65, n. 8

- MOORER, J.A., 1978 a, "The use of the linear prediction of speech in computer music applications", rapport IRCAM n.6/78

- MOORER, J.A., 1978 b, "How does a computer make music", Computer Music Journal (CMJ), vol. 2, n. 1

- MOORER, J.A., 1979, "About this reverberation business", CMJ, vol. 3, n. 1

- MOORER, J.A., CHAUVEAU, A., ABBOTT, C.W., EASTTY, P.C., LAWSON, J.R., 1979, "The 4C machine", CMJ, vol. 3, n. 3

- ORR, T., THOMAS, D.W., 1973, "Electronic sound synthesis", Wireless World, pag. 366-372, 429-434, 485-490 (august, september, october)

- PERONI, B., 1973, "Comunicazioni elettriche", Siderea, Roma

- PREVOT, Ph., 1980, "A.R.T.S. - A Real Time System: un système temps réel, conversationnel, pour piloter la machine 4C", rapport interne IRCAM

- PREVOT, Ph., 1981, "Controllo in tempo reale di un sistema di trattamento/sintesi del suono", Atti del 4. colloquio di informatica musicale, CNUCE, Pisa

- PRIEBERG, F.K., 1960, "Musica ex machina. Über das Verhältnis von Musik und Technik", Verlag Ullstein, Berlin

- RABINER, L., GOLD, R., 1975, "Theory and applications of digital signal processing", Prentice-Hall, Englewood Cliffs

- RISSET, J.C., 1965, "Situation de la recherche en acoustique musicale aux USA", bulletin du GAM n. 14

- RISSET, J.C., 1969, "L'ordinateur comme instrument de musique", bulletin du GAM n. 45

- RISSET, J.C., MATHEWS, M.V., 1969, "Analysis of musical instrument tones", Physics Today, vol. 22, n. 2

- ROADS, C., 1979, "A tutorial on non-linear distortion or waveshaping synthesis", CMJ, vol. 3, n. 2

- ROZENBERG, M., 1980, "Linear sweep synthesis and Moebius sounds", à paraître dans CMJ

- SAUNDERS, S., 1977, "Improved FM audio synthesis methods for real- time digital music generation", CMJ, vol. 1, n. 1

Bibliographie

- SCHAEFFER, P., 1952, "A la recherche d'une musique concrète", Paris

- SCHROEDER, M.R., 1962, "Natural-sounding artificial reverberation", JAES, vol. 10, pag. 219-223

- SUEN, C.Y., 1970, "Derivation of harmonic equations in nonlinear circuits", JAES, vol. 18, n. 6

- THOMAS, J.B., 1969, "Statistical communication theory", J. Wiley & sons, New York

- TRW, 1977, "TRW product note: LSI multipliers MPY-8AJ, MPY-12AJ, MPY-16AJ", TRW Inc., One Space Park, Redondo Beach, California

- VERCOE, B., 1971, "The Music 360 language for sound synthesis", Proceedings of the American Society of University Composers, n. 6, pag. 16 - 21

- VERCOE, B., 1980, "Music 11", Cambridge, Massachusetts