THESE de DOCTORAT D'ETAT

ès Sciences Mathématiques
(option Informatique)

présentée

à l'Université Pierre et Marie Curie
- Paris 6 -

par M. **Philippe Matherat**

pour obtenir le grade de DOCTEUR ès SCIENCES

Sujet de la thèse:
Contribution à l'augmentation de puissance des architectures de visus graphiques.

soutenue le 11 mai 1988

devant le jury composé de:

M. Jacques Arsac (Président)
M. Francis Devos
M. Jean-Jacques Lévy
M. Gérard Noguez (Rapporteur)
M. Claude Puech (Rapporteur)
M. Patrice Quinton (Rapporteur)

Résumé

La motivation de ce travail est la réalisation de circuits permettant d'afficher rapidement des images sur un écran d'ordinateur. Voici dix ans, nous avons proposé un circuit LSI, prenant en charge la gestion d'une mémoire d'image et l'écriture rapide de segments de droite et de caractères, dans une optique de "terminal graphique". Nous avons ensuite cherché à augmenter les performances de cette architecture et à l'adapter à l'environnement "station de travail". Nous sommes aujourd'hui convaincu que la solution ne passe pas par des circuits spécialisés, mais par la définition d'opérateurs généraux de calcul très puissants. Pour expliquer cet itinéraire, nous décrivons une suite d'expérimentations réalisées, précédée par une histoire des architectures de visualisation.

Mots clés

Architecture des ordinateurs, Visualisation graphique, Synthèse d'image, Circuits intégrés, Stations de travail, Co-processeurs graphiques, Parallélisme massif.

Remerciements

Ce travail a été réalisé au Laboratoire d'Informatique de l'Ecole Normale Supérieure, dirigé successivement par Jacques Arsac puis Claude Puech. Ils ont tous deux fait vivre cette structure dans laquelle j'ai pu imaginer ces expérimentations et rédiger ce texte.

Jean-Jacques Lévy m'a fait connaître l'activité de l'équipe des Laboratoires Bell qui a conçu le terminal Blit (et les conclusions associées sur l'implémentation du "BitBlT") au moment où je remettais en cause la volonté de notre équipe de "câbler" un maximum de primitives graphiques. Le cours que nous avons ensuite monté ensemble sur ce sujet m'a aidé pour faire avancer la rédaction de ce texte.

Gérard Noguez, avec qui j'ai entamé l'étude des contrôleurs d'écrans graphiques à l'occasion de mon stage de DEA, Patrice Quinton, dont les remarques très positives après sa lecture du manuscrit m'ont encouragé, ainsi que Francis Devos me font l'honneur de participer au Jury.

Sylviane Coué et Jacques Beigbeder m'ont suggéré de nombreuses corrections d'orthographe, de style et de typographie.

Je suis redevable à tous les membres du Laboratoire d'Informatique de l'ENS, avec lesquels il m'est agréable de travailler, et en particulier à ceux de l'équipe d'Architecture des Ordinateurs.

Une pensée particulière pour Sylviane qui m'a soutenu pendant la rédaction.

Chapitre 0: Introduction générale

A l'origine de ce travail, nous nous sommes posé la question:

"Quelle doit être l'architecture des circuits permettant d'afficher rapidement des images sur un écran d'ordinateur?"

Si l'on ne précise pas davantage la nature et la complexité des images souhaitées, on couvre tout le spectre des puissances de calcul. Où s'arrêter - et pourquoi s'arrêter?

Dans les années 60, cette question a visé l'affichage de caractères alphanumériques pour remplacer les télétypes, puis le tracé des segments de droite pour remplacer les tables traçantes à plume. La vitesse de tracé obtenue a permis l'animation. On a ensuite cherché davantage de réalisme. Les algorithmes modernes de synthèse d'image obtiennent ce réalisme en incluant, dans la description des scènes, de plus en plus de modèles physiques décrivant les phénomènes à visualiser.

Mais, entre-temps, sont apparus de nouveaux types d'images. Les "fenêtres" et les "pop-up menus" dessinés sur les écrans des micro-ordinateurs actuels n'existaient pas en 1970, et ne sont pas le résultat d'une recherche de réalisme au sens de l'image de synthèse. Cette façon de gérer un écran graphique, basée sur la généralisation de la primitive "BitBlT", est issue de la rencontre entre la recherche d'une meilleure interactivité et l'utilisation des circuits des années 75-80. Le succès de ces concepts est tel qu'aujourd'hui tout nouvel ordinateur doit être efficace dans la gestion du BitBlT.

Cette évolution dans la nature même des images souhaitées montre que la question initiale ne peut être traitée avec une méthodologie naïve du genre: étant donnée une liste d'images type, cherchons les circuits qui peuvent les calculer le plus vite possible.

Les arguments guidant les choix d'implémentation sont liés à des coûts de réalisation, pour des projets s'étalant sur plusieurs années, et utilisant des technologies évoluant très vite. Les facteurs économiques et sociaux y entrent pour une large part, et il est bien difficile d'aborder tout cela d'une façon satisfaisante dans le cadre d'un travail scientifique, c'est-à-dire principalement de formaliser les questions et d'obtenir des résultats quantitatifs.

Cette difficulté se rencontre d'ailleurs à chaque fois que l'on se pose la question: "ordinateur général ou circuiterie spécialisée?". Ceci sous-entend une évaluation par comparaison entre:

- d'une part un ordinateur général activé par un logiciel d'application,
- d'autre part la circuiterie spécialisée activée par un logiciel adapté.

Les deux logiciels sont bien sûr différents. En outre, ce que l'on cherche à calculer est mal connu. Dans le meilleur des cas, il est connu au départ mais évolue en cours de projet.

Voici dix ans, nous avons proposé un circuit LSI spécialisé, prenant en charge la gestion d'une mémoire d'image et l'écriture rapide de segments de droite et de caractères alphanumériques, dans une optique de "terminal graphique" bon marché. Depuis nous avons cherché à augmenter les performances de ce genre de dispositifs. Nous avons également cherché à l'adapter à l'environnement "station de travail".

Nous sommes aujourd'hui convaincu que la solution ne passe pas par des circuits spécialisés, mais par la définition d'opérateurs généraux de manipulation et de calcul sur une mémoire -comme pour beaucoup d'autres problèmes- avec toutefois la contrainte particulière (mais est-ce une contrainte?) de la petite taille (dimensions physiques de la machine).

Pour expliquer cet itinéraire, et cette conviction actuelle, nous avons choisi de présenter une suite d'expérimentations, précédée par une histoire des visus et la description des architectures ayant marqué les 15 dernières années.

Le plan de ce texte est donc le suivant:

Un premier chapitre rassemble les points communs à toutes les unités de visualisation, les problèmes à résoudre, et ce qui est imposé par la technologie des écrans, ceci fixant le cadre des réflexions qui suivront.

Les premières réalisations ont reconnu le segment de droite comme primitive essentielle -ceci étant lié à l'utilisation des visus comme terminaux- ce qui a conduit aux co-processeurs graphiques spécialisés, voie qui, à notre avis, s'achève vers 1980-1982. Le deuxième chapitre montre l'état d'esprit de ces architectures et leur évolution.

L'activité entre 1973 et 1983 de l'équipe de Xerox-PARC a bouleversé la conception des ordinateurs et des écrans, en inventant l'ordinateur individuel ainsi que la gestion des écrans par BitBlT. Nous décrivons ces machines dans le troisième chapitre, et nous montrons comment ces concepts, au départ si opposés aux architectures commerciales à base de microprocesseurs, ont pu se rencontrer dans une première machine tel que le Macintosh d'Apple, et s'imposer de façon générale.

Le quatrième chapitre décrit une réalisation personnelle dans laquelle nous avons tenté, dans les années 80-84, d'explorer systématiquement les architectures à base de co-processeur graphique, en recherchant une intégration maximum.

Le cinquième chapitre décrit rapidement les architectures des stations de travail contenant des accélérateurs de synthèse d'images réalistes, dont les résultats sont impressionnants et en évolution rapide, mais qui nous semblent limités par leur spécialisation.

Le sixième et le septième chapitre décrivent notre vision actuelle d'une machine de synthèse d'image puissante. D'une part la visualisation proprement dite consiste en un DMA relativement classique, quoique très rapide, entre la mémoire générale et l'écran. D'autre part, la synthèse est réalisée sur une machine générale massivement parallèle, intimement imbriquée avec la mémoire, et d'une dimension physique compatible avec un ordinateur individuel.

Avertissement concernant l'utilisation de termes anglo-saxons

Nous nous sommes efforcé de n'utiliser que des termes français pour la rédaction de ce texte. Néanmoins, certaines expressions nous ont parues difficiles à traduire sans perte importante de sens, ou de facilité d'expression. Nous les avons donc conservées dans leur langue d'origine, et nous les avons citées, suivies chacune d'une définition, dans un glossaire placé à la fin.

La langue parlée dans les laboratoires de recherche en Informatique est un jargon incorporant de façon très critiquable de nombreux termes anglais ayant des équivalents français répandus. Mais, ce n'est pas le cas de tous les nouveaux termes. Les chercheurs lisent des publications dont une grande majorité est rédigée de l'autre côté de l'Atlantique. L'introduction de concepts nouveaux demande la formation de néologismes. On apprend alors en même temps le concept et le mot pour le désigner. Nous pensons que dans ce cas précis, la recherche d'un soi-disant "équivalent" français (deuxième néologisme pour le même concept) ne comporte que des inconvénients. Cela augmente la difficulté de compréhension du concept et affaiblit le sens des nouveaux mots. Que penserions nous si nos ancêtres avaient refusé les mots arabes "zéro" ou "chiffre"? Peut-être la xénophobie anti-américaine est-elle plus forte aujourd'hui?

Chapitre 1: Introduction aux architectures de contrôleurs graphiques: historique, balayage cavalier et balayage télévision, contraintes temporelles

Historique: balayage cavalier (Random-scan)

Les premiers écrans graphiques sont apparus dans les années 60 [NS79], et sont dérivés des techniques utilisées pour les oscilloscopes: la déflection du faisceau d'électrons est réalisée par des plaques créant un champ électrostatique (figure 1.1).

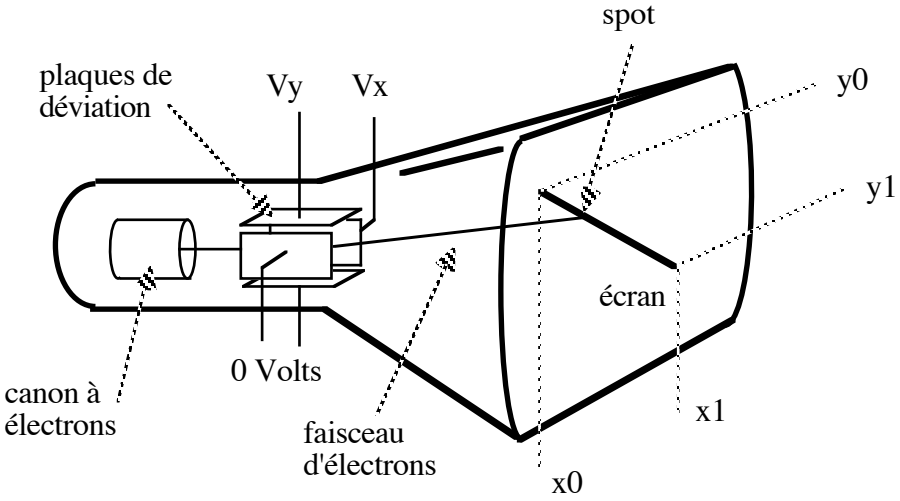

Fig. 1.1: Balayage cavalier (Random-scan)

Deux tensions électriques de déviation V_x et V_y définissent l'adresse (x, y) du spot, point d'impact du faisceau sur l'écran. Pour tracer le segment de (x_0, y_0) à (x_1, y_1), on doit créer les tensions de la figure 1.2. Toute l'image est construite ainsi, en concaténant les déviations élémentaires correspondant à chacun des segments de droite (vecteurs) de l'image.

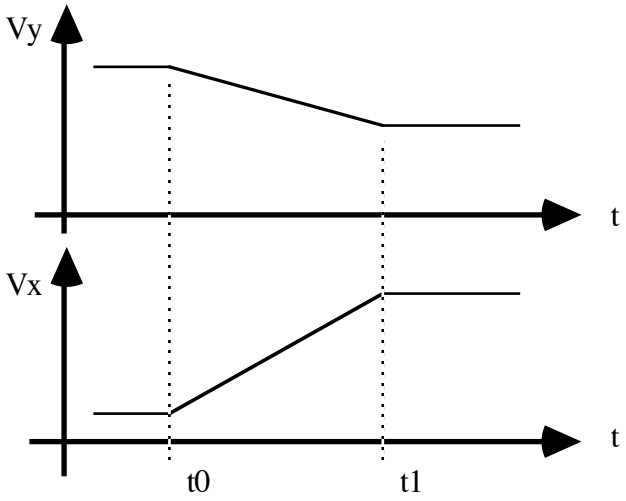

Fig. 1.2: Tensions de déviation

Le spot suit donc le chemin correspondant à la figure à tracer, exactement comme la plume d'une table traçante. Ce mode de déviation est appelé "balayage cavalier", par référence au déplacement de cette pièce dans le jeu d'échec.

La télévision et son balayage systématique de l'écran existait (depuis les années 30), mais elle n'a pas été utilisée pour l'affichage graphique avant 1975 environ. La principale raison est qu'elle nécessite une mémorisation complète de l'image, ce qui n'a été possible qu'avec des circuits intégrés de mémoire suffisamment denses, suffisamment rapides, et suffisamment bon marché.

Une difficulté liée au balayage cavalier est la nécessité d'une grande précision absolue de la déviation du spot. En effet, si l'on dessine sur l'écran la figure 1.3, il se peut que la croix d'une part et le cercle d'autre part soient tracés à des instants éloignés, ceci étant dû à l'organisation de la base de données des éléments à afficher. Entre-temps, le spot a pu s'éloigner notablement. Il faut une bonne répétitivité de la déviation pour que les positions respectives de la croix et du cercle soient correctes. En particulier, une dérive lente des composants électroniques (lente par rapport au tracé d'un vecteur) peut créer un défaut important.

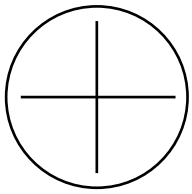

Fig. 1.3: Précision absolue

Ce besoin de précision absolue de la déviation oblige à utiliser la déviation électrostatique, qui est plus chère que la déviation magnétique utilisée en télévision. Les rampes des tensions de déviation de la figure 1.2 doivent être précises (linéaires) et sont générées par une électronique digitale (à base de compteurs), suivie de convertisseurs digitaux / analogiques. La vitesse de variation des tensions Vx et Vy étant limitée par la qualité de l'électronique et par la vitesse des électrons dans le faisceau, on utilise le plus souvent une description relative des vecteurs (ΔX,ΔY), à partir de laquelle on fait évoluer X et Y à vitesse constante grâce à une électronique digitale appelée "générateur de vecteurs" (figure 1.4).

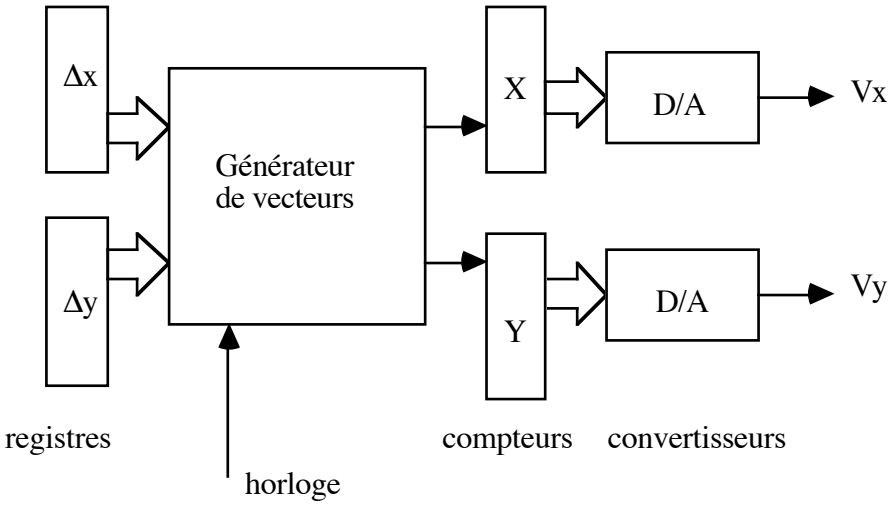

Fig. 1.4: Génération des tensions Vx et Vy

Toutes les images sont donc construites à partir de "vecteurs". On doit tracer une suite de "petits vecteurs" pour afficher des caractères ou pour remplir une surface. Ceci donne un aspect "fil de fer" aux images.

L'algorithmique est adaptée à cette description en vecteurs. Pour les scènes en perspective, on transforme les extrémités des vecteurs. On calcule des "*lignes* cachées", et non des faces cachées. On fait du "clipping" par dichotomie sur un segment de droite [SS68] [SSS74] [SH74].

Les algorithmes implémentés dans les générateurs de vecteurs sont dérivés de ceux utilisés pour les tables traçantes [Bre65] [Pit67] [Den73] [KR75] [Hor76] [Bre77].

Une particularité importante est la nécessité de "rafraîchir" l'image. En effet, pour pouvoir afficher des images en mouvement, on ne peut pas utiliser des phosphores à extinction lente. Il faut donc retracer l'image périodiquement (environ 50 fois par sec.), suffisamment souvent pour qu'elle ne clignote pas. Comme la vitesse de tracé des vecteurs est bornée, la longueur totale du trajet du spot pour décrire une image qui ne clignote pas est aussi bornée. La complexité des images affichables est donc limitée.

Le séquencement des opérations d'affichage est effectué par un "processeur d'affichage" [MS68] [NS79] qui:

- charge les registres ΔX et ΔY,
- contrôle l'allumage et l'extinction du faisceau (pour déplacer éventuellement le spot sans tracer),
- lance le tracé par le générateur de vecteurs,
- incrémente l'adresse mémoire où il va chercher ΔX et ΔY,
- et boucle.

L'interface entre le processeur hôte et ce processeur est donc une mémoire contenant des listes de vecteurs à tracer. Outre des déplacements ΔX et ΔY, on place dans cette mémoire des instructions de saut permettant:

- de définir la fin de l'image et le rebouclage au début,
- de structurer l'image en sous-images,
- de déplacer une sous-image sur l'écran simplement en modifiant un vecteur éteint.

Les images sont donc des listes chaînées de vecteurs. Une sous-image est appelée "segment d'image" (à ne pas confondre avec un segment de droite qui est la primitive vecteur).

Les écrans graphiques à balayage cavalier les plus répandus ont été l'IBM 2250 et le DEC 340.

Pour animer des images complexes telles que visualisation de scènes en perspective (simulateurs de vol par exemple), on augmente la complexité du processeur d'affichage (figure 1.5), qui devient un "pipe-line" effectuant en série sur les vecteurs, à partir d'une base de données 3D, les opérations suivantes:

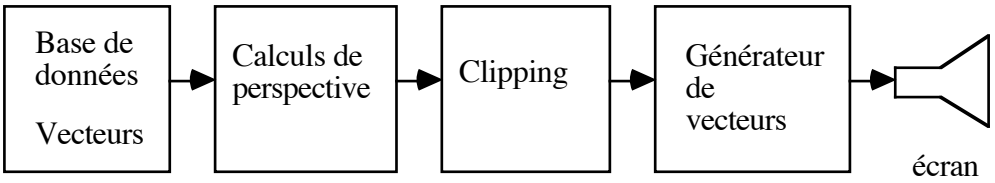

Fig. 1.5: Pipe-line d'animation 3D

- transformation de coordonnées pour calculs de perspective (multiplication de matrices 4 x 4 [Sut68]),
- clipping pour garder les vecteurs ou portions de vecteurs visibles [SS68],
- génération de vecteurs.

Une telle machine a été réalisée dès 1968 par Evans & Sutherland: le LDS-1. Cette machine ne comporte pas de mémoire intermédiaire dans le pipe-line et ne traite pas les lignes cachées. Bien d'autres ont succédé. Ce pipe-line a été réutilisé, mais transformé pour des polygones, dans le cas des écrans à balayage télévision [Cla80a].

Affichage en couleurs

Ces écrans sont en général monochromes. La couleur est possible avec les "tubes à pénétration". Dans ces tubes, la surface interne de l'écran est recouverte de 2 couches de luminophores, l'une verte, l'autre rouge. En ajustant la tension d'accélération des électrons, donc leur énergie d'impact, ceux-ci pénètrent plus ou moins dans ces couches. On obtient 4 couleurs possibles: vert, jaune, orange, rouge. Comme cette tension d'accélération est THT (très haute tension - autour de 15 kV), sa vitesse de variation possible est faible par rapport à la vitesse de variation des tensions de déflection. On trace donc tous les vecteurs d'une couleur donnée, puis on change la tension d'accélération des électrons, puis on trace tous les vecteurs de la deuxième couleur, etc. Les défauts de précision de déviation se traduisent alors par des décalages entre les parties d'une couleur et les parties d'une autre couleur, ce qui est difficile à éviter complètement, vu l'aspect délicat de l'électronique analogique mise en jeu et les interactions entre circuits.

Ecrans Tektronix à mémoire

En 1968, Tektronix a présenté un écran à mémoire bon marché, qui a eu un grand impact commercial, et a banalisé la fonction "terminal graphique". Dans cet écran, le luminophore mémorise l'image, qui est entretenue par un faisceau d'électrons "secondaires" qui "arrose" en permanence toute la surface de l'écran. Il n'est plus nécessaire de rafraîchir l'image. On peut donc afficher des images sans limite de complexité (autre que celle du temps d'affichage). Le revers de la médaille est l'impossibilité de l'effacement sélectif d'une partie de l'image. La seule possibilité est l'effacement total, qui prend un temps de l'ordre de une seconde et qui est incompatible avec l'interactivité. Néanmoins, ces terminaux (4010, puis 4014) ont été très répandus, jusque vers 1980. L'absence de processeur d'affichage et de mémoire d'image sous forme de liste de vecteurs permettait le faible coût. L'utilisation se faisait via une ligne asynchrone à 9600 bauds, éventuellement 1200. Sur cette ligne, l'ordinateur hôte envoie des coordonnées absolues (x,y) découpées pour être transmises sous forme de codes caractères ASCII, précédés par des caractères "Escape".

Le succès de ces appareils a entraîné le développement de logiciels à base de tracé de vecteurs, comme PLOT10 ou GKS.

Avec ces appareils -et leur connexion à un hôte par une ligne lente- apparaît un besoin nouveau: celui de coder des sous-ensembles d'images pour condenser l'information transmise. Ce besoin de condensation de l'information sur la ligne a imposé la notion de "primitive" graphique, qui a pris une grande importance dans la structuration des logiciels et qui subsiste dans les esprits, malgré le remplacement des "terminaux" par les "stations de travail graphiques".

Ecrans à balayage télévision (Raster-scan)

Dans les écrans à balayage télévision, la position du spot est imposée par un balayage systématique, indépendant de l'image à afficher (figure 1.6) [CL76]. Le spot balaye de gauche à droite, et plus lentement de haut en bas. Il suffit de 2 oscillateurs générant des signaux "triangulaires" (figure 1.7).

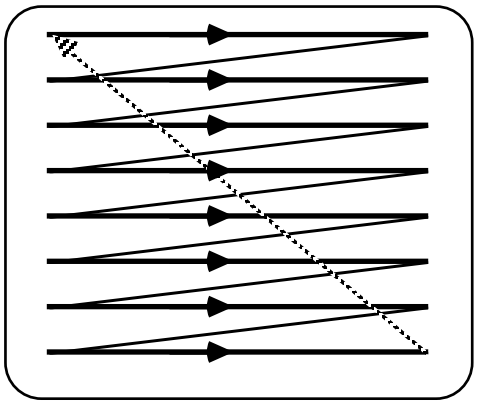

Fig. 1.6: Balayage TV (Raster-scan)

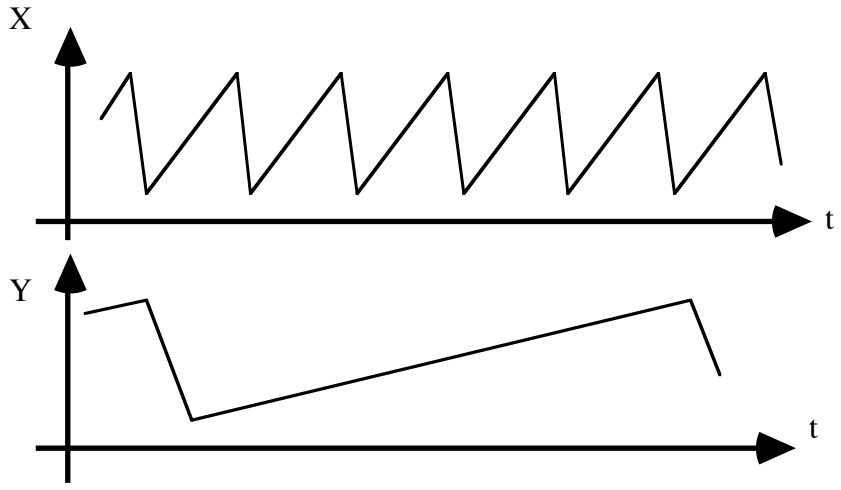

Fig. 1.7: Courants de déviation

L'angle que font les lignes vidéo avec l'horizontale est de 1 / 625, ce qui est négligeable. De toutes façons, le calage des bobines de déviation (voir plus loin) se fait en

fonctionnement, un opérateur regardant une mire affichée sur l'écran, et le résultat est d'une précision moindre.

La qualité demandée à ces signaux de déviation est beaucoup moins grande que dans le balayage cavalier. En effet, si une dérive lente de la position se produit, elle affectera la position globale de l'image, et non des éléments particuliers, pourvu que les signaux de déviation soient périodiques, ce qui est simple du point de vue électronique.

Il est tout de même nécessaire que les rampes de montée de ces signaux soient droites (linéarité de l'image), mais avec une précision de quelques centièmes seulement, ce qui est relativement classique (charge d'un condensateur à courant constant).

Ceci permet d'utiliser une déviation magnétique et non électrostatique. On crée un champ magnétique proportionnel au courant de déviation par 2 bobines ("de Helmholtz" - figure 1.8). Ces bobines sont extérieures au tube cathodique -ce qui simplifie la réalisation- alors que les plaques électrostatiques étaient intérieures. Elles sont déformées pour venir épouser la forme du col du tube.

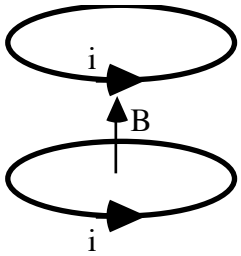

Fig. 1.8: Bobines de Helmholtz

Ces techniques sont celles de la télévision, développées dans les années 30. Les fréquences de déviation sont relativement faibles pour la télévision proprement dite (standard 625 lignes):

- 50 Hz de fréquence trame,
- 15 kHz de fréquence ligne,

plus critiques pour une visu 1000 x 1000:

- jusqu'à 70 Hz de fréquence trame,
- jusqu'à 80 kHz de fréquence ligne.

Cette vitesse de déviation est donc liée à la résolution de l'image, mais pas à sa complexité (au sens des objets affichés).

L'ensemble {écran, bobines de déviation, oscillateurs (ligne et trame)} constitue le "moniteur vidéo" (figure 1.10).

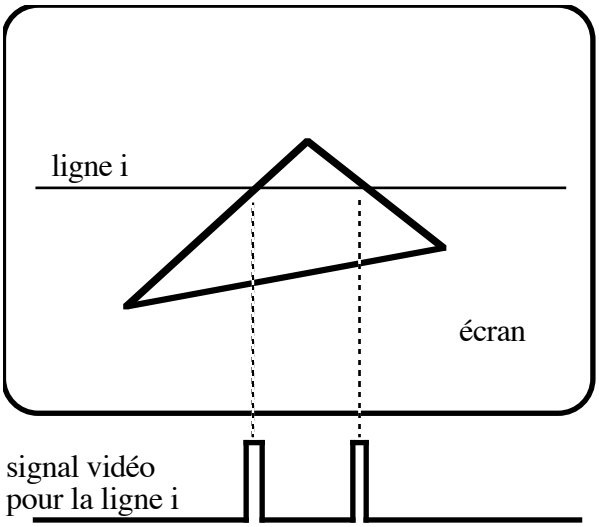

Fig. 1.9: Signal Vidéo

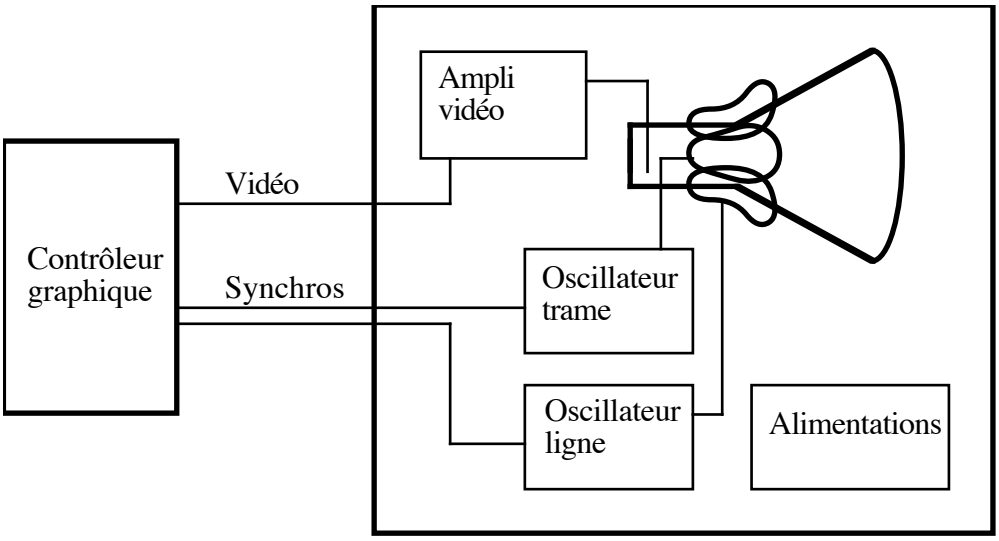

Fig. 1.10: Moniteur Vidéo

L'image est formée par la modulation de l'intensité du faisceau d'électrons (figure 1.9). Le signal qui module cette intensité et qui porte donc l'information de l'image, est appelé "signal vidéo". C'est un signal logique sur un bit en N&B, analogique sur un fil si

on affiche des niveaux de gris, analogique sur 3 fils si on affiche des couleurs (RVB: rouge, vert, bleu). Dans ce dernier cas, le tube comporte 3 canons à électrons, et 3 sortes de luminophores, chacun ne "voyant" qu'un seul canon à travers une grille métallique (shadow-mask).

Ces signaux vidéo sont définis en liaison avec des signaux de synchronisation (figure 1.11) générés également par le contrôleur graphique, et sur lesquels le moniteur "cale" ses oscillateurs (à l'aide de circuits "boucle à verrouillage de phase" par ex.). Les oscillateurs ligne et trame fonctionnent même en absence de signaux de synchronisation, mais dès que ceux-ci sont présents, les oscillateurs se calent en fréquence et en phase. Le déphasage (figure 1.11) n'est pas critique. Il correspond à une translation de l'image sur l'écran. Il est réglable par un bouton du moniteur.

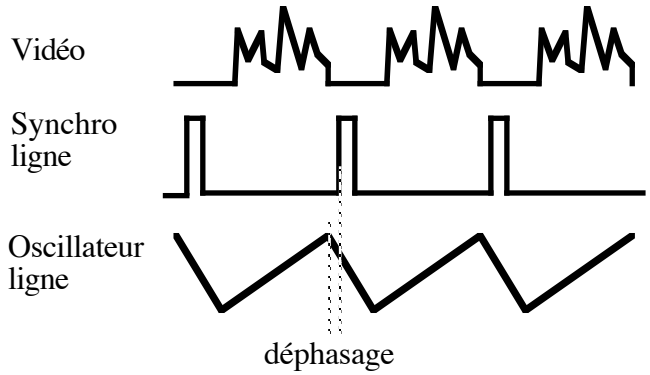

Fig. 1.11: Synchronisation

Regardons la fréquence de succession des points le long d'une ligne:

- télévision: 15 kHz x 800 points = 12 MHz (80 ns par point),
- écran 1000 x 1200: 80 kHz x 1200 points = 100 MHz (10 ns par point).

La valeur élevée de la fréquence de succession des points sur le signal vidéo, ainsi que l'ordre, indépendant de la succession des objets, montrent la difficulté de passer de cette description sous forme d'objets à celle des points sur l'écran. Il est nécessaire de mémoriser "quelque part" tous les points de l'écran dans une "mémoire d'image". Historiquement, cela a d'abord été fait avec des disques ou des tambours magnétiques, puis des registres à décalage CCD. Aujourd'hui, l'emploi de mémoires à semi-conducteurs à accès aléatoire s'est généralisé.

La visualisation est une lecture séquentielle et périodique de la totalité de la mémoire. L'écriture utilise l'accès aléatoire imposé par les algorithmes de synthèse.

De cette organisation est née la notion de "pixels" (contraction de "picture-element"), qui est le point d'image correspondant à une adresse dans la mémoire d'image. Cette notion ne pouvait exister sur un écran à balayage cavalier, et n'était pas connue en télévision où le signal vidéo est continu le long d'une ligne.

C'est la réalisation de cette mémoire qui explique l'apparition relativement tardive de cette technique (vers 1975). Ceci peut se comprendre en comparant l'évolution de la technologie des mémoires et son adaptation progressive aux besoins des écrans (tableau ci-dessous).

années	intégration	128 x 128	256 x 256	512 x 512	1024 x 1024
1971	1 K / circuit	16 circuits	64 circuits		
1974	4 K / circuit	4 circuits	16 circuits	64 circuits	
1976	16 K / circuit		4 circuits	16 circuits	64 circuits
1979	64 K / circuit			4 circuits	16 circuits

Les avantages de cette technique par rapport au balayage cavalier sont les suivants:

- pas de limitation de la complexité de l'image affichée,
- pas de limitation du nombre de couleurs différentes possibles,
- lien avec la vidéo: caméra, magnétoscope.

Différents problèmes liés au moniteur et au balayage

Scintillement de l'image lié à la fréquence trame

Le choix des molécules pouvant constituer les luminophores est assez limité. Ces molécules sont caractérisées par:

- la couleur de la lumière émise,
- le rendement (lumière émise / énergie des électrons reçus),
- la vitesse d'extinction.

L'extinction se fait suivant une exponentielle décroissante dont on ne considère généralement que la constante de temps. Pour le phosphore blanc le plus commun, elle est de l'ordre de 40 μs, soit moins que le temps d'une ligne télévision (64 μs). Ce qui signifie que sur un écran de télévision, une seule ligne émet de la lumière à un instant donné! C'est l'oeil qui assemble les lignes pour constituer l'image. On peut utiliser des phosphores plus lents -il en existe ayant une constante de temps d'extinction de 150 ms- mais ce n'est pas compatible avec les images en mouvement.

On doit donc afficher les images avec une fréquence trame supérieure à une fréquence caractéristique de l'oeil (appelée par définition "persistance rétinienne"). Mais ceci est assez complexe à en juger par le fait que les connaissances sur l'oeil évoluent davantage avec l'utilisation de ces écrans que l'inverse.

Dans l'affichage pour le cinéma, le problème est très différent. L'obturateur est un disque tournant (figure 1.12). Le dispositif de déplacement du film est irrégulier car l'image doit être fixe quand elle est projetée. Il faut donc déplacer le film quand le disque obture la lumière.

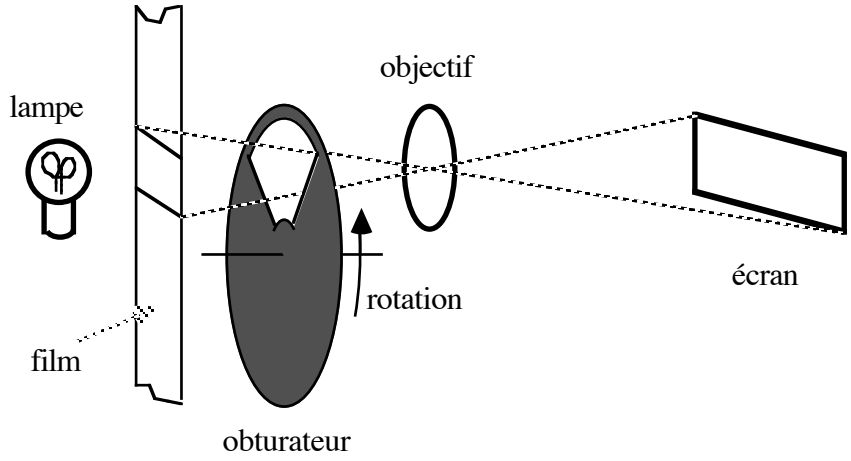

Fig. 1.12: Projection cinéma

Les difficultés de réalisation de la mécanique, et les problèmes de longueur du film, ont imposé au début du siècle une fréquence de 24 images par seconde. Ce nombre est suffisant si on considère la quantité d'information projetée: un mouvement rapide est suffisamment bien décomposé pour le cerveau, à cette fréquence. Par contre, une fréquence de 24 éclairs par seconde est insuffisante pour l'oeil et donne une impression de clignotement insupportable. On utilise donc un disque à 2 fenêtres (si le disque fait un tour par image). On a donc 24 images par seconde, et 48 éclairs. Chaque image est vue 2 fois. Le film doit être déplacé en un temps inférieur à 10 ms (mais ceci est lié en outre à l'angle d'ouverture des fenêtres du disque qui est un autre paramètre, lié au compromis entre temps d'ouverture et temps de déplacement du film).

On en a déduit à l'époque que le temps de persistance rétinienne était compris entre 1 / 24^{e} et 1 / 48^{e} de seconde. Il est à noter que:

- rien n'est rémanent dans le système d'affichage, mais seulement dans la rétine,
- quand l'image est sur l'écran, elle est complète.

Quand les normes de télévision ont été mises au point, 3 contraintes s'imposaient:

- minimiser toutes ces fréquences pour un problème de transmission: minimiser la largeur de bande HF utilisée,
- être compatible avec le cinéma,
- utiliser des fréquences sous-multiples de la fréquence d'alimentation secteur pour des problèmes d'électronique (50 Hz en France, 60 Hz aux USA).

On a choisi 25 images par secondes, mais avec 50 trames. Ceci est possible par la méthode de "l'entrelacement" qui consiste à afficher une trame constituée uniquement des lignes impaires, puis la suivante constituée des lignes paires.

Plusieurs problèmes sont nouveaux par rapport au cinéma:

- pour une image issue d'une caméra vidéo, et qui utilise obligatoirement le même entrelacement, si l'image évolue, les lignes paires et impaires ne s'assemblent pas, car correspondent à 2 images différentes (il y a en fait 50 demi-images par secondes, toutes différentes),
- l'oeil doit, non seulement suppléer par sa rémanence aux instants sans image, mais il doit en outre assembler les images ligne par ligne puisqu'il ne reçoit de la lumière que d'une ligne à la fois. Ceci donne l'effet (plus ou moins consciemment ressenti suivant les individus) d'une vague de lumière qui descend périodiquement l'écran,
- du point de vue de l'information affichée, il n'y a que 25 images par seconde. Si dans une image, un objet n'appartient qu'à une trame (reflet sur une arête horizontale, nombreuses dans un escalier par exemple), cet objet va clignoter à 25 Hz. Il y a là à considérer la cohérence entre les lignes, qui est liée à la quantité d'information réelle contenue dans l'image.

Si on ne s'occupe plus de télévision, mais d'affichage graphique pour ordinateur, et que l'on ne cherche plus de compatibilité avec le balayage télévision, plusieurs contraintes disparaissent:

- le débit de transmission HF,
- la compatibilité avec le cinéma,
- la compatibilité avec le secteur, problème lié à la qualité des alimentations (qui n'est d'ailleurs plus un problème sur les téléviseurs de qualité),
- les problèmes de caméra.

Par contre, on va afficher des images n'ayant que rarement une cohérence ligne (on affiche souvent des lignes horizontales). L'entrelacement est alors inutile, et n'apporte que des complications. On a le choix de la fréquence trame (égale à la fréquence image s'il n'y a pas d'entrelacement). On est donc amené à augmenter cette fréquence pour le confort de l'oeil. On s'est aperçu alors que les connaissances sur l'oeil étaient à réviser, car la fréquence de 50 Hz est trop faible. On augmente ce chiffre constamment (actuellement, de nombreux écrans utilisent 66 Hz). Il y a toujours des problèmes liés aux mouvements rapides de l'oeil (saccades) qui rompent l'assemblage de l'image sur la rétine, et qui ne peu-

vent se résoudre que par des phosphores très rémanents. Un compromis est à trouver avec le débit demandé aux mémoires d'image.

Correction "de gamma"

Le rendement de l'ensemble {canon à électron, phosphore} n'est pas linéaire. Ce qui revient à dire que la relation entre la tension électrique qui commande le canon à électrons et la lumière que reçoit l'oeil n'est pas une proportion constante. Cette relation, caractéristique du moniteur, se modélise assez bien par une exponentielle. La tradition veut qu'on nomme "gamma" l'exposant et "correction de gamma" la correction à apporter au signal vidéo pour obtenir un bon rendu de l'image.

En télévision, cette correction est effectuée avant l'émission radio et ne doit donc pas être prise en charge par le moniteur (ceci est rendu possible par une normalisation sur les tubes cathodiques). En synthèse d'image, il est nécessaire d'effectuer cette correction pour les images réalistes demandant un bon rendu des valeurs (en particulier si l'on traite l'anti-aliasing).

Anti-aliasing

L'utilisation d'écrans vidéo, associée au codage dans une mémoire d'image, ont fait apparaître la notion de "pixel", liée à la représentation d'une image comme un tableau de nombres. Ceci n'existe ni en photographie, où la position des molécules de colorants est aléatoire, ni pour le balayage cavalier dans lequel l'image n'est jamais décomposée en tableau de points.

Dans le cas général (image quelconque), on doit admettre que le processus de synthèse revient à un échantillonnage d'une image idéale, décrite mathématiquement, qui se heurte aux mêmes difficultés que la prise de vue par une caméra CCD dont les capteurs sont disposés sur une grille, et ce problème est la version à 2 dimensions d'un problème bien connu en acquisition d'un signal analogique: "l'aliasing".

Les méthodes pour éviter ce problème sont donc nommées "d'anti-aliasing".

Le défaut le plus rapide à constater est que les traits dessinés sur une grille présentent des "marches". Si l'on ne considère qu'un trait isolé, on peut croire que ce défaut peut être

diminué en augmentant la résolution, mais dès qu'on étudie un réseau de traits, on comprend que ça ne suffit pas: les décrochements des traits forment des alignements qui créent des "moirages" (figure 1.13); dans le cas d'un réseau de traits serrés (distance voisine de 1 pixel), l'image obtenue peut être complètement différente (rotation du réseau de traits) [Cro77] [Cro78].

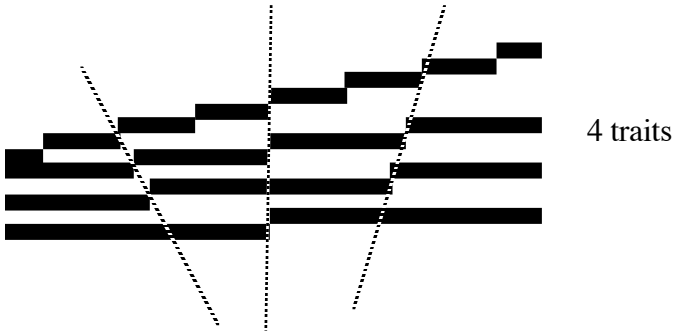

Fig. 1.13: Moirage

Un autre type de défauts se produit lorsque des objets ont une taille voisine de la distance inter-pixel. Considérons le cas du calcul d'une perspective. Celle-ci peut rendre très petits des objets qui ne se trouvent alors sur aucun point de l'écran (figure 1.14).

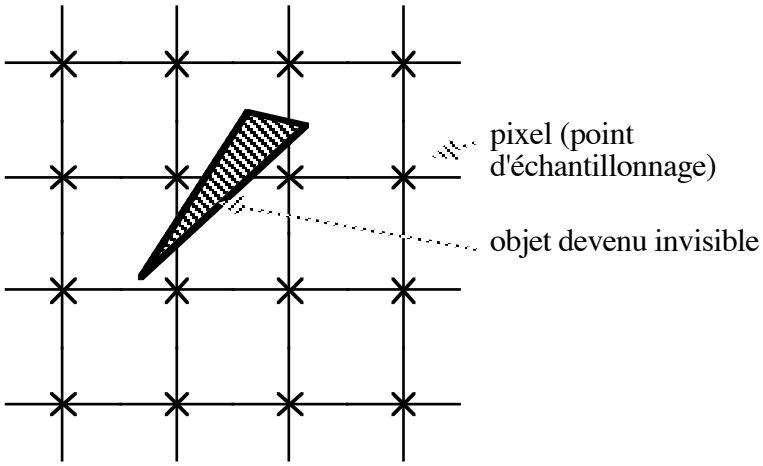

Fig. 1.14: Echantillonnage spatial

Pourtant, un objet très petit peut avoir beaucoup d'importance pour la compréhension de l'image, par exemple dans le cas d'un phare qui doit être visible de très loin. Si on associe un mouvement à ce calcul de perspective, on peut voir ces points lumineux clignoter. Par exemple, la simulation d'une piste d'atterrissage de nuit est catastrophique car les balises clignotent de façon apparemment désordonnée.

On s'aperçoit facilement qu'il ne suffit jamais d'augmenter la résolution pour résoudre ces problèmes. Leur origine est liée à l'échantillonnage. Dans tous les cas cités, on cherche à échantillonner une grandeur à 2 dimensions qui contient des fréquences spatiales supérieures à la fréquence des pixels. Le résultat n'est pas que la disparition de fréquences spatiales élevées, mais l'apparition de fréquences basses ("aliases"), comme on peut s'en convaincre rapidement dans le cas à une dimension de la figure 1.15. Le signal recueilli est obligatoirement de fréquence inférieure à la fréquence d'échantillonnage et liée à la différence entre les fréquences présentes et la fréquence d'échantillonnage (repliement de spectre). Une augmentation de la fréquence d'échantillonnage ne suffit pas tant que subsistent des fréquences supérieures, ce qui est toujours le cas pour une image de synthèse où les objets sont définis par leurs coordonnées mathématiques.

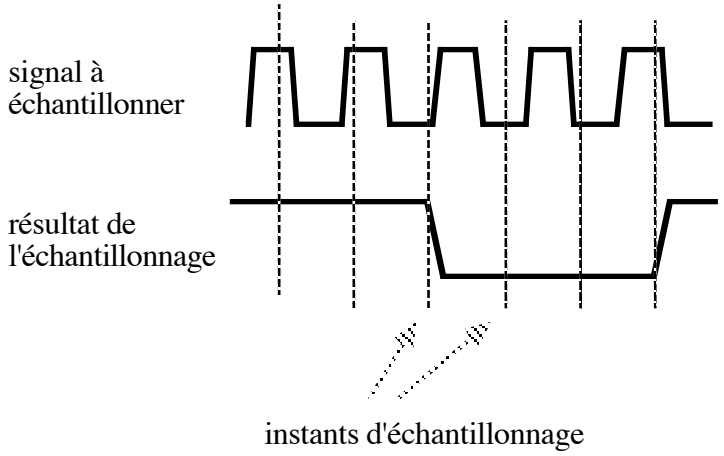

Fig. 1.15: Apparition de fréquence basse

La seule solution est de filtrer les fréquences élevées (supérieures à 1/2 de la fréquence des pixels) avant d'échantillonner (i.e. avant de calculer la valeur du pixel).

Dans le cas de la synthèse d'image, ce filtrage peut s'effectuer, pour beaucoup d'applications, par une convolution entre une fonction de diffraction et la fonction à afficher. Ces 2 fonctions ne sont calculées avec une résolution supérieure à la résolution de l'écran que pour certains pixels et ne nécessitent donc pas trop de calculs supplémentaires [Wei81] [Cat78].

Une application importante de ces méthodes de filtrage est la conséquence pour l'affichage des caractères alphanumériques, car dans un texte imprimé ceux-ci sont disjoints et la convolution peut être précalculée.

Souvent, on affiche les caractères comme des matrices de points allumés et éteints (figure 1.16). Il est clair qu'on ne peut déplacer sur l'écran de tels caractères d'une distance égale à un demi-pixel sans le déformer complètement. Ainsi, une visu graphique qui affiche 80 caractères par ligne ne peut en afficher 81.

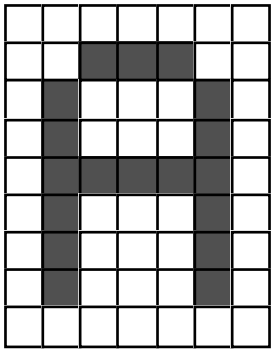

Fig. 1.16: Caractère alphanumérique

Et pourtant, si on digitalise le signal vidéo d'une caméra télévision pointée vers une image imprimée, avec une même fréquence de pixels, mais 4 bits par pixel, le passage de 80 caractères par ligne à 81 ne pose souvent aucun problème (changement de 1/80 ième de la qualité). Pourtant, dans ce cas, l'image est aussi digitalisée, à la même fréquence, et rien n'empêche de la synthétiser par programme.

L'explication est que, la plupart du temps, la diffraction par l'objectif de la caméra fait le travail de convolution mentionné plus haut. Il ne faut pas choisir un "trop bon" objectif, sinon les défauts apparaissent. Dans ce cas, on peut utiliser un filtrage électronique sur le signal vidéo, mais ceci n'est efficace que dans la direction horizontale.

Ceci conduit à l'idée d'un générateur de caractères calculant ce filtrage. Les expériences réalisées dans ce domaine [KU81] [War80] montrent qu'avec une matrice de 5 x 8 pixels et 4 bits par pixel, la qualité des caractères est supérieure à celle obtenue par la méthode traditionnelle, et qu'on peut représenter des fontes de caractères multiples, avec des facteurs de zooms non entiers, des rotations d'angles quelconques, etc...

L'application est bien sûr le traitement de textes puisque sur un écran 800 x 500 on peut représenter une feuille de papier format A4 destinée à être imprimée sur une imprimante laser. La qualité est inférieure à celle de l'imprimante mais la mise en page peut être strictement identique [Kay77].

Il est ainsi peu utile d'afficher une image de plus de 1000 x 1000 (ce qui correspond par ailleurs au pouvoir séparateur de l'oeil!). Une telle image, pour être exploitable, doit être

examinée à l'aide d'un zoom (un vrai, pas un grossissement des pixels). Si l'on souhaite examiner une image à la fois localement et globalement (par exemple un plan de circuit intégré de 5 mm x 5 mm précis au micron), on peut représenter l'image globale dégradée dans une fenêtre, et l'image locale très agrandie dans une autre.

Malgré le caractère déjà ancien de ces expériences, de tels écrans sont peu répandus vu le supplément de coût.

Double-buffering

Nous avons parlé de la mémoire d'image, lue périodiquement pour la visualisation, et modifiée parallèlement par le processeur d'écriture.

Cette mémoire peut se trouver en mémoire centrale de l'ordinateur ou dans une mémoire spécialisée.

Indépendamment de cet aspect architecture, ce buffer accédé par 2 processeurs concurrents est souvent doublé. Ceci est avantageux dans le cas de figures en mouvement. En effet, si l'on souhaite créer une image entièrement nouvelle à chaque trame, il faut dessiner la prochaine image dans un buffer différent de celui de visualisation et permuter les buffers en fin de trame.

Dans le cas d'animations plus simples, on peut n'utiliser qu'un seul buffer. Deux cas sont possibles:

- soit la modification de l'image est relativement lente par rapport à la fréquence trame. C'est le cas sur les écrans "bit-map" des micro-ordinateurs ou des stations de CAO. On modifie l'image pendant la visualisation. Un objet ajouté ou enlevé peut n'être présent que partiellement pendant une image. L'effet peut ne pas être trop désagréable, mais ceci dépend fortement du type d'interaction,
- soit on s'arrange pour ne modifier l'image que lors du temps de retour de l'oscillateur vertical. Les objets sont donc toujours complets. Les inconvénients sont liés au temps relativement court disponible pour modifier l'image (de l'ordre du 1/10 ième du temps total), et à la difficulté de gérer alors les autres éléments "temps réel" de l'interactivité (clavier, souris), surtout dans le cas d'un système d'exploitation qui le permet mal (comme UNIX).

La palette et les convertisseurs digitaux / analogiques (DAC)

Un grand nombre de problèmes de l'affichage graphique plaident pour un opérateur de transcodage très paramétrable avant l'entrée du moniteur vidéo.

Le principal argument est la nécessité de banaliser complètement le codage des informations dans la mémoire d'image:

- une application souhaitera des niveaux de gris subtilement nuancés, une autre cherchera plutôt des couleurs variées,
- une application codera une image sur tous les bits disponibles, une autre stockera simultanément plusieurs images différentes, chacune sur moins de bits,
- de nombreux algorithmes codent autre chose que l'intensité lumineuse dans la mémoire d'image: profondeur z (troisième dimension) pour l'élimination de parties cachées, buffers de variables intermédiaires diverses,
- rendu particulier des couleurs pour un algorithme d'anti-aliasing.

Plutôt que de concevoir une mémoire d'image spécialisée pour chacune de ces applications, il est préférable de laisser l'utilisateur (le programmeur) disposer à son gré des b bits par pixel. Ceci impose un transcodage très paramétrable sur la sortie vidéo [JG78].

Une autre catégorie d'arguments est celle liée aux différences entre les moniteurs. Nous avons déjà signalé la non-linéarité des canons à électrons. Cette courbe de réponse change en fonction du moniteur.

Inversement, l'existence de ce transcodage rapidement modifiable apporte des applications nouvelles, par exemple l'animation de l'écran sans modification de la mémoire d'image [Sho79].

La méthode de réalisation de ce transcodage est souvent l'utilisation d'une mémoire à double accès située entre la mémoire d'image et le moniteur (figure 1.17). Il faut avoir à l'esprit 2 caractéristiques de cette mémoire:

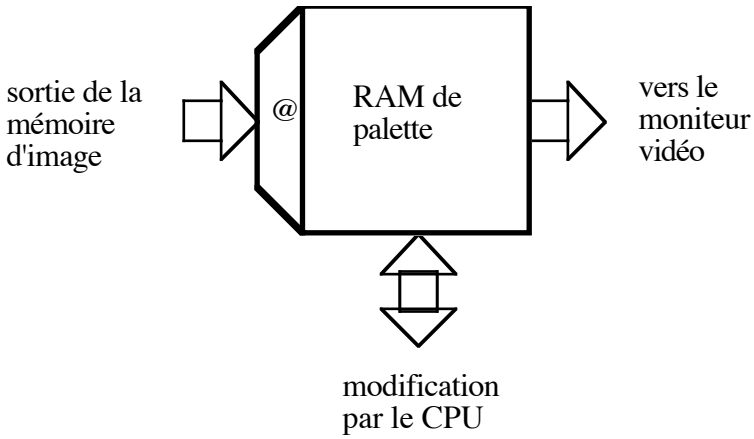

Fig. 1.17: Palette

- elle est très rapide puisque accédée à la fréquence des pixels sur le signal vidéo,
- la capacité est 2^b mots.

Ces contraintes ne posent pas de problème pour des applications du type 500 x 500, b = 4 (16 mots accédés à 25 MHz), mais deviennent très dures pour des résolutions élevées et un grand nombre de bits par pixels. Alors, deux techniques peuvent être utilisées: il faut soit pipe-liner beaucoup cette mémoire, soit perdre un peu de généralité en la découpant en plusieurs "banques". En effet, si 2^{24} est gigantesque, 2^{12} est compatible avec la réalisation par une mémoire. On peut par exemple rajouter un opérateur de type "crossbar" pour répartir les sorties de la mémoire d'image sur les 2 mémoires de transcodage.

Nous n'avons pas parlé de l'entrée des moniteurs. De ce côté, les choix sont plus restreints car on se limite à 8 bits pour coder le signal attaquant un canon à électrons. On estime en effet que, si on appelle D la dynamique des tubes réalisable, la sensibilité de l'oeil étant une variation de 2%:

$$(1{,}02)^n = D$$

d'où l'on déduit que n maximum est de l'ordre de 160 [Cat79]. La puissance de 2 immédiatement supérieure est 256.

En couleur, il y a 3 canons à électrons: R, V, B, ce qui fait 24 bits à l'entrée du moniteur. Ceci pousse en général à disposer de 3 tables de transcodage sortant chacune des mots de 8 bits.

Le double accès de ces mémoires ne pose pas de problème, car il n'est nécessaire d'en modifier le contenu que pendant le retour de l'oscillateur trame, temps pendant lequel le signal vidéo ne porte pas d'information utile.

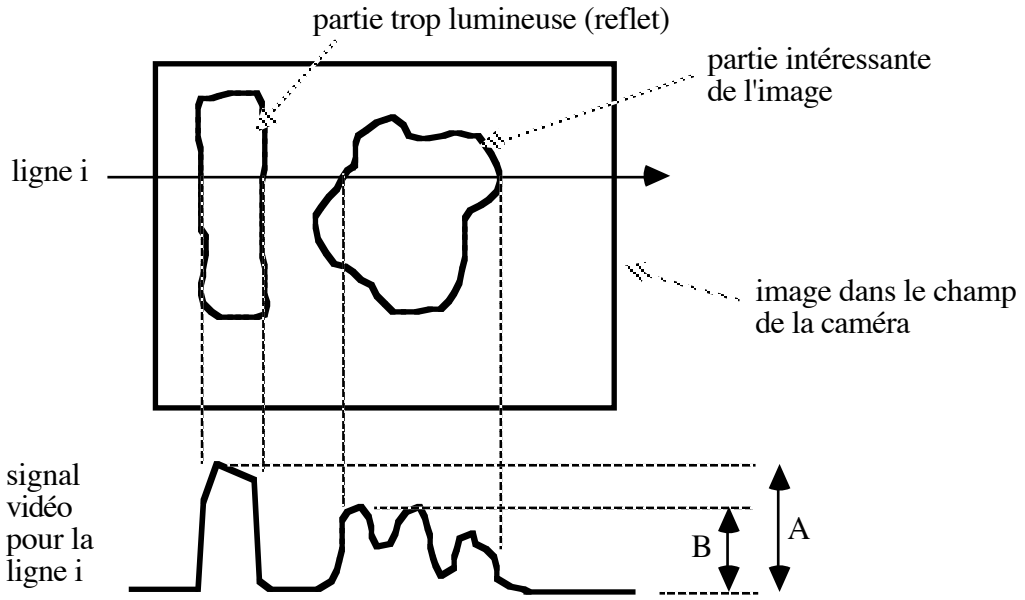

Fig. 1.18: Normalisation du signal vidéo

Il faut signaler qu'un dispositif analogue est utilisé pour la digitalisation d'une image caméra vidéo (figure 1.19). Ceci permet d'adapter la plage et la précision de digitalisation pour obtenir le maximum d'information utile (figure 1.18). Le signal S est, par construction de la caméra (contrôle automatique de gain de l'amplificateur vidéo), toujours cadré entre 0 et 1 Volt (plage A). Les 2^n niveaux du convertisseur sont donc équi-répartis sur la plage A. On peut choisir comme on le désire une répartition des 2^m niveaux de la mémoire sur la plage B. Ceci condense la quantité d'information dans la mémoire d'image et peut accélérer les traitements sur l'image.

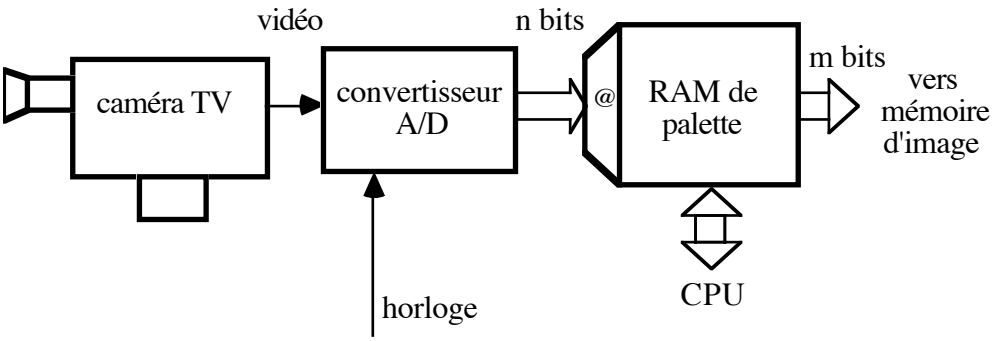

Fig. 1.19: Acquisition d'image

Chapitre 2: Les contrôleurs graphiques télévision orientés "vecteurs"

Présentation

Actuellement les écrans graphiques utilisent largement la primitive BitBlT, particulièrement adaptée à la manipulation de fenêtres sur l'écran et à l'affichage de caractères alphanumériques de qualité.

Mais les écrans graphiques à balayage télévision ont d'abord utilisé la primitive "vecteur", suite à l'influence des écrans à balayage cavalier qui l'avaient utilisée exclusivement. Ainsi, pendant longtemps, *graphique* a été lié à *vecteur* et opposé à *alphanumérique*.

Cette opposition remonte à la période antérieure à l'existence des écrans, où les sorties d'ordinateur sur papier étaient:

- soit des machines à écrire pour des caractères alphanumériques,
- soit des tables traçantes à plumes pour les graphiques.

Quand les écrans de visualisation apparaissent, ils reprennent cette séparation, principalement pour la raison que les écrans de visualisation alphanumériques sont moins complexes. En effet, un tableau de 24 lignes de 80 caractères se code dans une mémoire de 2 K octets et est simple à afficher si on ménage un espacement égal entre les caractères, comme dans les anciennes machines à écrire.

On a donc dans les années 70 (si l'on excepte les machines BitBlT), 2 sortes d'écrans de visualisation:

- des écrans uniquement alphanumériques, où les caractères sont situés sur une grille [Gas77] [GB77]. Le balayage est systématique (télévision), mais on ne mémorise pas les pixels. L'électronique est simple, les pixels n'existent qu'au tout dernier moment de la génération du signal vidéo, grâce à une mémoire morte (appelée "générateur de caractères") qui contient la forme d'une fonte 5 x 7 ou 7 x 9 maximum.

- des écrans dits "graphiques", dont la primitive de base est le vecteur. Ces écrans permettent souvent aussi d'afficher des caractères, par succession de petits vecteurs. Au début des années 70, ils sont très chers s'ils sont à balayage télévision car les mémoires à semi-conducteurs permettant de stocker tous les pixels ne sont pas denses. Ceci explique la grande durée de vie des consoles Tektronix de la série 4010 (de 1968 jusque vers 1980), bon marché, qui ont été à la base de la démocratisation des écrans graphiques. Ces machines à balayage cavalier et à écran à mémoire remplissaient la fonction terminal (voir figure 2.1) via une ligne de débit moyen (9600 ou 1200 bauds, voire même 300 par le réseau commuté).

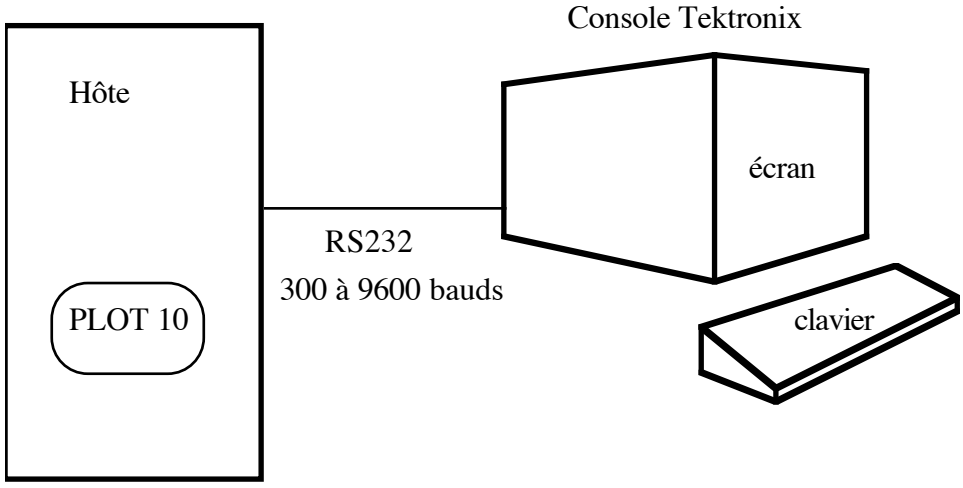

Fig. 2.1: "Terminal" graphique

Ce qui passe sur la ligne est une succession de codes de caractères, compatibles avec le code ASCII des terminaux alphanumériques. On a seulement rajouté quelques codes (séquences commençant par "Escape") permettant de changer de mode de fonctionnement. Dans le mode "graphique", l'hôte envoie un couple (x_i, y_i), et le terminal déplace le spot du point précédent (x_{i-1}, y_{i-1}) au point indiqué. Les coordonnées x et y sont codées sur 10 bits chacune (12 dans les versions suivantes 4014). Ces 20 (respectivement 24) bits sont répartis sur 4 caractères (respectivement 5).

Ce mode d'échange avec un terminal graphique est devenu une norme de fait, ainsi que certaines librairies de logiciels aidant à construire ces dessins (la plus connue étant

PLOT10, suivie de CORE puis GKS): tracés et graduations d'axes, mises à l'échelle, histogrammes, camemberts, etc...

Le premier co-processeur graphique télévision: EF9365

Nous avons commencé l'étude de ce circuit en 1976 [Mat78b]. Des écrans graphiques à balayage télévision existaient, mais étaient chers. Le prix des mémoires MOS baissait et les premiers modèles utilisant l'adressage multiplexé venaient d'apparaître (Intel 2104, 4 K bits à 250 F pièce). Les microprocesseurs devenaient plus puissants (8 bits en 1974). Nous avons pensé qu'une visu graphique à balayage télévision pouvait être bon marché et remplacer les Tektronix (permettant en outre l'effacement sélectif et la couleur) à condition (voir figure 2.2):

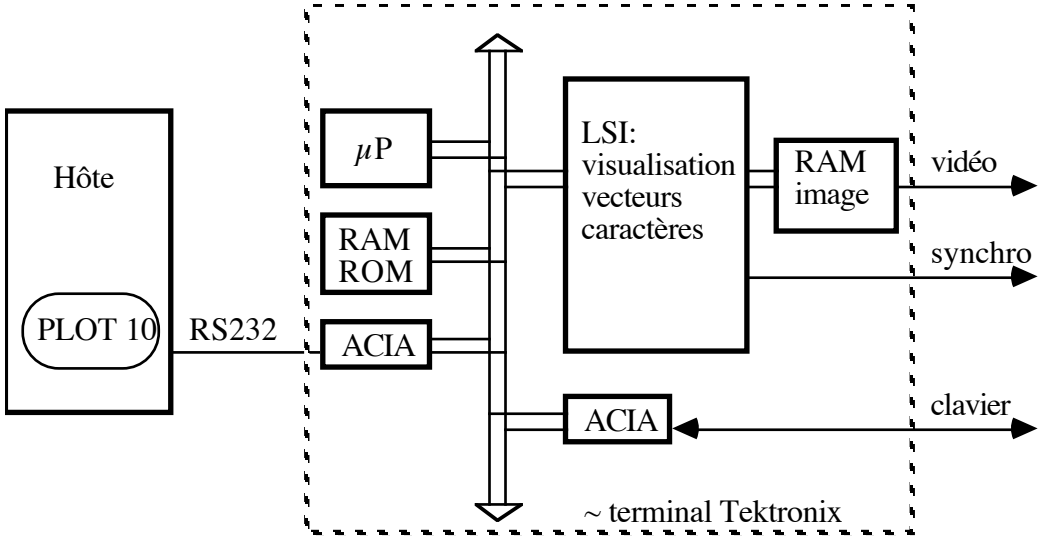

Fig. 2.2: Simulation de terminal

- de réutiliser un écran très répandu: la télévision classique 625 lignes, 25 Hz (entrelacé à 50 Hz),

- de condenser toute l'électronique (visualisation + générateur de vecteurs + générateur de caractères) en un circuit LSI,

- de mettre un microprocesseur 8 bits dans le terminal pour interpréter les codes type Tektronix et continuer ainsi à utiliser les logiciels genre PLOT10.

Raisons détaillées de certains choix

Pourquoi une mémoire d'image séparée de la mémoire du microprocesseur ?

- Une image 512 x 512 x 4 bits (16 couleurs) occupe 128 K octets alors que les microprocesseurs 8 bits adressent au maximum 64 K.

- Le débit demandé à la mémoire d'image uniquement pour la visualisation est déjà de 1,75 M octets/s pour une image monochrome, ce qui est supérieur au (ou voisin du) débit des bus modulaires de l'époque.

Pourquoi inclure un générateur de vecteurs câblé ?

- A l'époque, nous étions persuadés que le problème principal était de tracer les vecteurs le plus vite possible. Cela était hérité des machines à balayage cavalier où la vitesse du générateur de vecteurs limitait la complexité de l'image affichée.

- Les consoles Tektronix étaient utilisées en CAO, par exemple de circuits intégrés. Or un masque de circuit LSI, dont le codage pouvait comprendre de l'ordre de quelques dizaines de milliers de vecteurs, pouvait demander plusieurs minutes de tracé à l'écran (limité par le débit de la ligne à 9600 bauds, lui-même lié à l'architecture des Tektronix). On souhaitait davantage d'interactivité.

- Il devenait possible d'afficher des images animées grâce à l'effacement rendu plus facile, consistant uniquement en une mise à 0 des mémoires, ce qui peut se faire soit sélectivement pour un vecteur, soit sur toute l'image en couplant cela avec la visualisation dans des cycles de LME ("Lecture-Modification-Ecriture" ou RMW: "Read-Modify-Write"). Il faut alors être capable de dessiner une image nouvelle toutes les 20 ms. Si on double la vitesse du générateur de vecteurs, on doublera la complexité possible de l'image en mouvement.

- Les microprocesseurs avaient une puissance de l'ordre de 0,1 MIPS, soit une addition 8 bits en 2 à 4 μs. Cette puissance ne permet un tracé de vecteurs qu'en 10 à 20 μs par pixel minimum.

Structure du EF9365

Nous ne reprendrons pas ici tous les détails de l'architecture de ce circuit, décrit dans [Mat78a] [Mat78b] [Mat80], mais seulement les grandes lignes, pour en parler avec un éclairage plus actuel.

Visualisation

Le choix du balayage télévision français, et le choix d'un pixel "carré" imposent une fréquence pixel de 14 MHz (71 ns) pour 512 pixels par ligne.

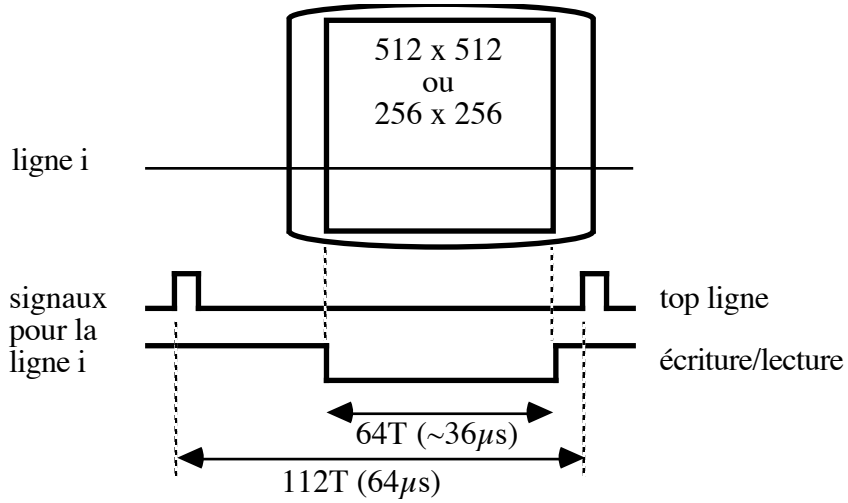

Fig. 2.3: Séquencement ligne du circuit EF9365

Les résolutions choisies étant carrées (256 x 256 ou 512 x 512), on obtient les temps de la figure 2.3. En organisant la mémoire en mots de 8 bits (figure 2.4), on obtient un temps de cycle de 570 ns (suffisant, mais sans largesse, pour les mémoires de l'époque). Les compteurs de visualisation se découpent en 2 parties (figure 2.5):

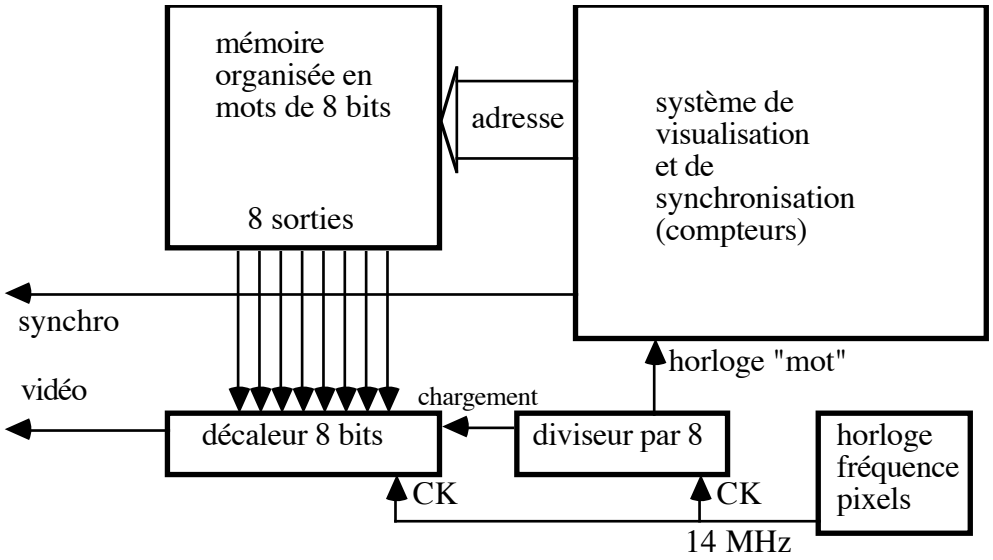

Fig. 2.4: Gestion mémoire du circuit EF9365

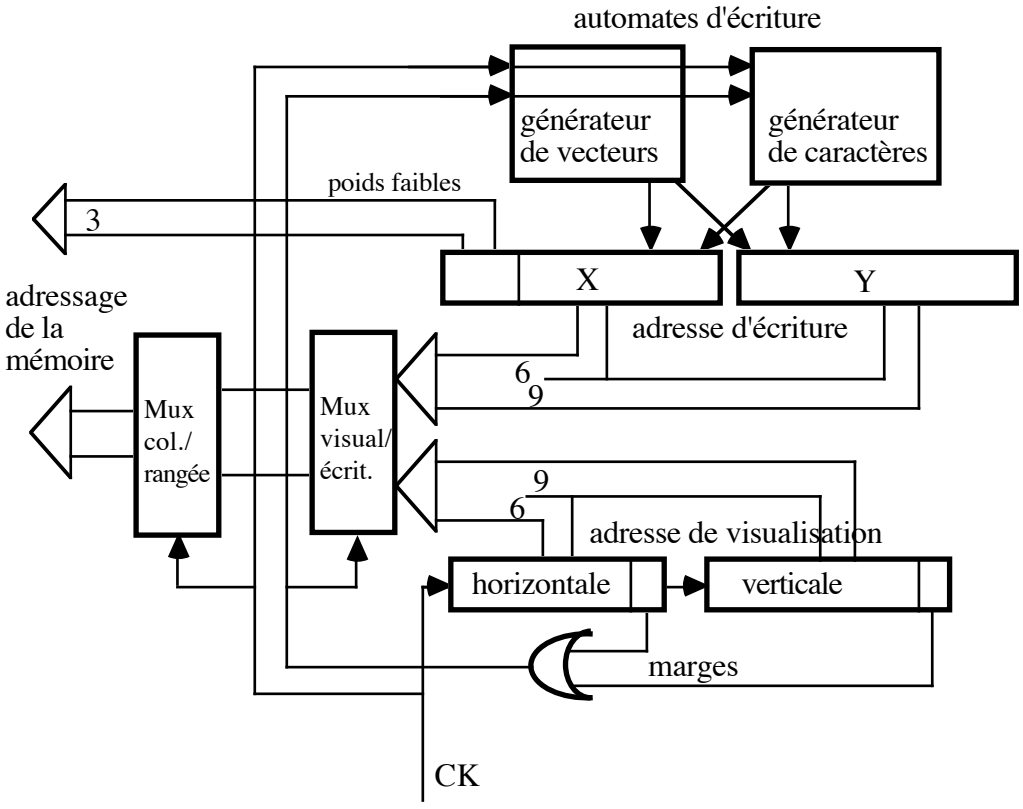

Fig. 2.5: Génération des adresses par le circuit EF9365

- le compteur horizontal qui séquence les cycles le long d'une ligne vidéo. Il comprend 7 bits (112 cycles par ligne) dont 6 servent directement d'adresse horizontale de visualisation (64 cycles), le 7 ième indiquant les marges droite et gauche,

- le compteur vertical qui séquence les lignes le long d'une image. Il comprend 10 bits (625 lignes) dont 9 servent directement d'adresse verticale de visualisation (512 lignes), le 10 ième indiquant les marges haute et basse. En réalité, ce compteur est un peu plus compliqué car l'entrelacement oblige à ce que le bit faible (de parité ligne) soit mis en poids le plus fort (de n° de trame). En outre, un paramétrage par un signal externe permet de fonctionner ou non en mode entrelacé.

Des reconnaisseurs sur des valeurs de ces compteurs génèrent le signal de synchronisation du moniteur télévision.

Lors des marges, l'accès aux mémoires est autorisé pour l'écriture, à raison de un pixel par cycle, grâce à un autre couple de registres d'adresse (registres X et Y), qui joue le rôle de "l'adresse du spot" d'un écran à balayage cavalier.

Sur la figure 2.5, on a représenté en bas les compteurs de visualisation, le bit de poids fort de chacun indiquant une marge, le OU de ces 2 fils permettant de commuter le multiplexeur visualisation / écriture et permettant le fonctionnement des automates d'écriture (générateurs de vecteurs et de caractères).

On a représenté le multiplexeur rangées / colonnes qui est imposé par l'utilisation de mémoires multiplexées. Les 3 fils de poids faibles de X (registre d'adresse écriture) sélectionnent un pixel parmi 8 d'un mot mémoire. Ils doivent être sortis par un port séparé car ils n'existent pas en visualisation.

Générateur de vecteurs

Le générateur de vecteurs est une boîte (figure 2.6) qui reçoit en entrée les projections du vecteur à tracer ΔX et ΔY et fabrique en sortie des signaux d'incrémentation et de décrémentation des registres d'adresse d'écriture X,Y (qui seront donc implémentés sous forme de compteurs Up/Down), au rythme du signal d'horloge CK.

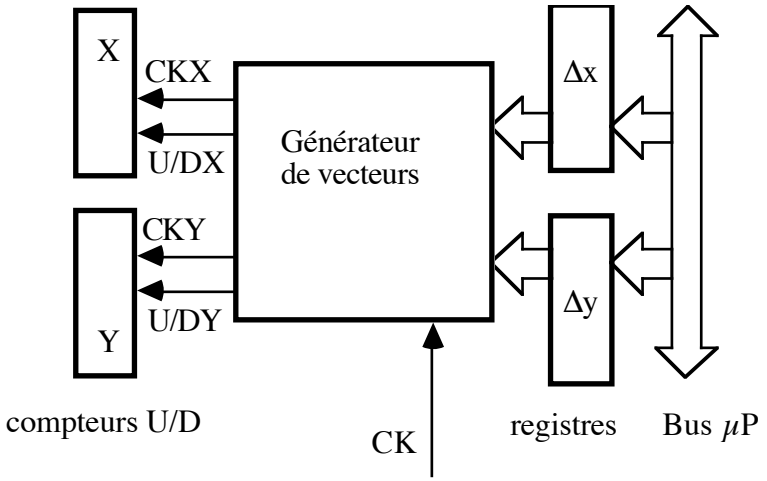

Fig. 2.6: Génération de vecteurs

Quelle structure choisir pour ce générateur de vecteurs ?

Flashback: l'algorithme de Bresenham

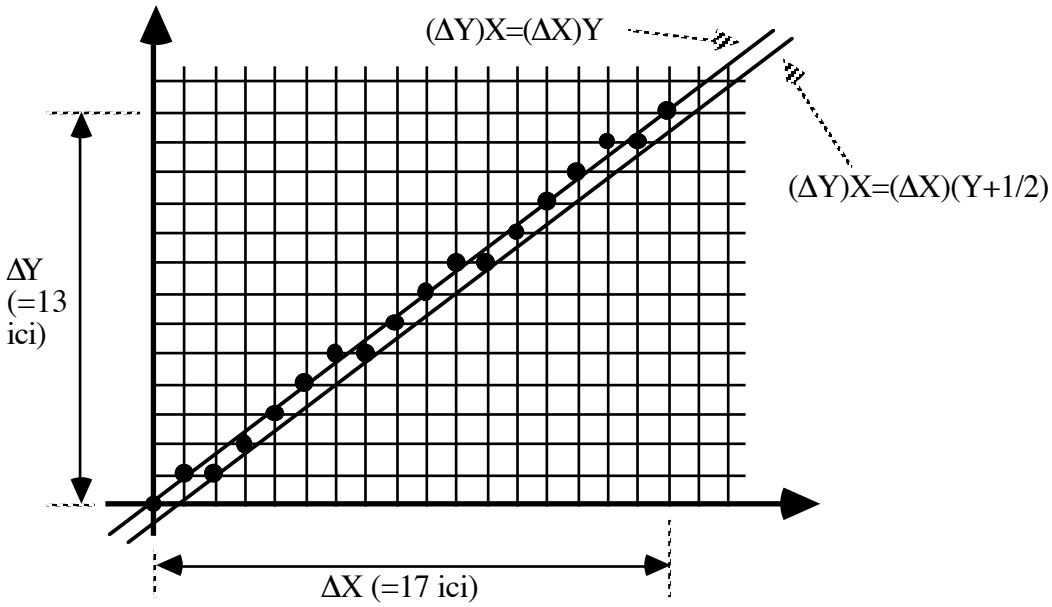

Fig. 2.7: Tracé de Bresenham

Cet algorithme, publié en 1965 et destiné à la commande d'une table traçante munie de moteurs pas-à-pas [Bre65], a eu beaucoup de succès pour les écrans "bit-map" et a été

généralisé pour toutes sortes de courbes [Bre77], car il repose sur des calculs ne comprenant que des additions entières.

On se restreint aux vecteurs d'origine (0,0) et situés dans le premier octant ($\Delta X > 0$, $\Delta Y \geq 0$, $\Delta X \geq \Delta Y$) comme dessiné figure 2.7.

Comme $\Delta X \geq \Delta Y$, on va modifier un seul pixel par colonne (par valeur de X). Comme on ne fait que augmenter Y, et que la pente est ≤ 1, alors à chaque colonne, Y est soit le même que dans la colonne précédente, soit augmenté de 1 au plus. Le critère pour augmenter Y est le franchissement de la droite d'équation:

$$s = (\Delta Y)\, X - (\Delta X)\, (Y+1/2) = 0$$

située 1/2 pixel plus bas que la droite $X / Y = \Delta X / \Delta Y$.

Or cette quantité s est facile à calculer, en partant de $-\Delta X / 2$ quand $X = Y = 0$, en additionnant ΔY systématiquement à chaque pixel, et en soustrayant ΔX si on décide d'augmenter Y (quand s deviendrait positif si on ne le faisait pas).

Le programme est donc:

```
début   x = 0 ; y = 0 ; s = -ΔX/2 ;
        tant_que   x ≤ ΔX   faire
                marquer_pixel(x,y) ; x = x+1 ; s = s+ΔY ;
                si  s ≥ 0  alors   y = y+1 ; s = s-ΔX  fsi
        fait
fin
```

Fin du flashback

Réalisation de notre générateur de vecteurs

Si on implémente cet algorithme de façon programmée ou microprogrammée, il est nécessaire de dérouler quelques instructions (3 au moins - X et Y sont des compteurs câblés) dans la boucle interne d'où:

- si on réalise un petit processeur général en MOS, la fréquence maximum possible à l'époque étant 2 MHz, on obtient un temps de tracé de quelques μs par pixel.

- si on utilise un μP en tranches (type Amd 2900), on peut aller au moins 3 fois plus vite, mais cela va à l'encontre de l'intégration en un seul LSI permettant de baisser les coûts du terminal.

Nous avons donc opté pour une solution entièrement câblée dans le but d'écrire un pixel par cycle du circuit (570 ns).

Le montage de la figure 2.8 est celui que nous avons utilisé, après quelques modifications, car il ne fonctionne pas tel quel (en particulier à cause de la boucle additionneur / multiplexeur).

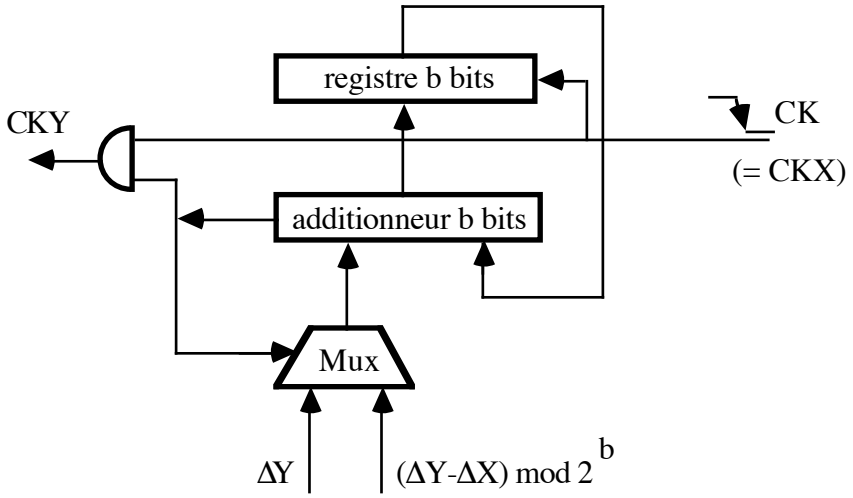

Fig. 2.8: Bresenham "câblé"

A l'origine, ce schéma était issu d'une réflexion sur les diviseurs de fréquence programmables, mais on peut remarquer que le contenu du registre évolue exactement de la même façon que la variable s du programme cité pour l'algorithme de Bresenham.

Clipping automatique

Dans notre implémentation, les registres X et Y sont de 12 bits chacun, alors que 9 suffiraient pour 512 x 512. Cela permet de traiter simplement un premier niveau de clipping. En effet, soient (figure 2.9) les 2 vecteurs A et B tracés dans le sens des flèches et enchaînés. Ces 2 vecteurs coupent le bord droit de l'écran. Si l'on ne disposait que de 9 bits pour X et Y, on devrait choisir entre:

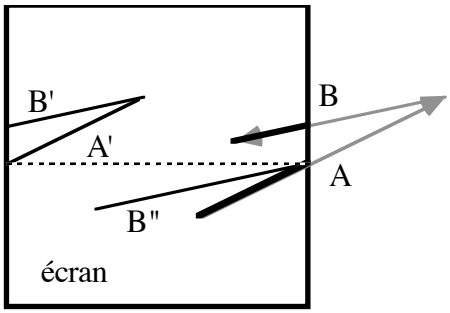

Fig. 2.9: Clipping automatique

- ignorer le débordement de X (écran torique), auquel cas on dessinerait A' et B',
- ou bloquer le comptage sur le bord, auquel cas on dessinerait B",

ce qui signifie que le logiciel de préparation des vecteurs serait obligé de prendre en charge ce premier niveau de clipping.

Générateur de caractères

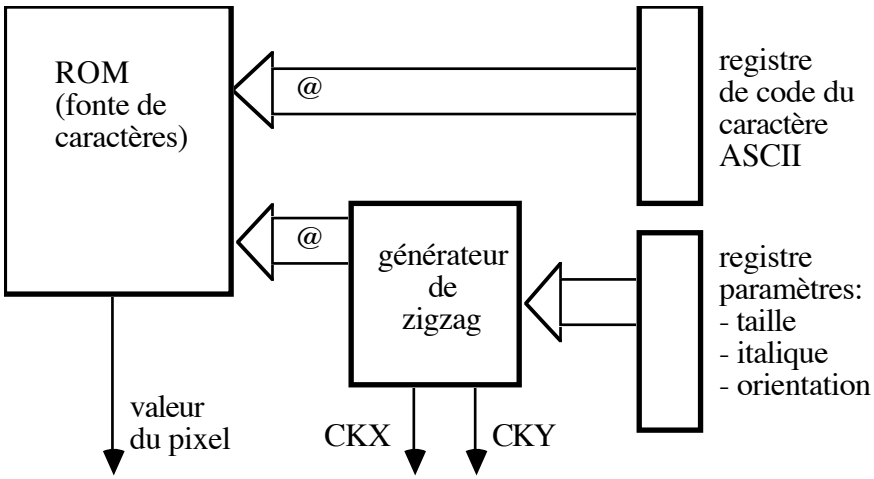

Fig. 2.10: Générateur de caractères

Le générateur de caractères est une boîte (figure 2.10) qui agit de même que le générateur de vecteurs par des incrémentations et décrémentations des compteurs X et Y d'écriture. Mais ici, au lieu de calculer le trajet (comme pour les vecteurs) et de marquer toujours de la même couleur les pixels atteints, on va fonctionner à trajet fixe, et marquer les pixels atteints en blanc ou en noir en fonction de la description du caractère contenue dans une mémoire (fonte). Le trajet doit permettre de balayer tout le rectangle représentant

le caractère. Ce rectangle peut être dilaté (taille des caractères), incliné (italique), ou tourné de 90° (écriture de bas en haut). Le trajet choisi est un zigzag qui permet d'optimiser le temps de tracé (penser que pour modifier "l'adresse du spot" de plus de un pixel en un cycle, il faudrait un additionneur supplémentaire).

Critique de ce circuit

Ce circuit a eu beaucoup de succès (près d'un million d'exemplaires vendus) car il permet de fabriquer très simplement une visu graphique à balayage télévision suivant le schéma de la figure 2.2. Ce genre de visu est un terminal qui réalise un sur-ensemble des fonctions du terminal Tektronix et peut donc s'adresser au même marché, en offrant les fonctions supplémentaires de l'effacement sélectif et de la couleur. C'est la voie dans laquelle nous avons engagé par exemple la société SECAPA dès l'apparition du circuit.

Performances, utilisation en terminal

Le temps de tracé d'un vecteur est de 570 ns par pixel en pointe (lors des marges) et de 1,3 μs en moyenne (en tenant compte de la proportion des cycles non occupés par la visualisation). Cela donne 700 μs pour un vecteur plein écran. C'est une vitesse très correcte, qui permet entre autres de tenir la simulation Tektronix à 19200 bauds, ce qui n'aurait pas été possible avec un générateur de vecteurs 2 fois plus lent, car dans le code Tektronix, un vecteur long peut être déclenché par seulement 2 caractères.

En animation, si l'on utilise 2 buffers image, on peut tracer 20 ms / 1,3 μs soit environ 15000 pixels par trame, soit 150 vecteurs de 100 pixels, ce qui, bien que moyen, n'était pas possible sur un Tektronix bas de gamme mais seulement sur le modèle à 600 kF.

Utilisation en micro-ordinateur

Dans l'utilisation en terminal, les événements sont rythmés par l'arrivée des commandes sur la ligne, ce qui n'est plus le cas internement à un micro-ordinateur où les commandes de vecteurs sont générées localement. Dans ce cas, on ne peut plus ignorer "l'overhead" du logiciel de préparation des vecteurs:

- même si les vecteurs (couples $\Delta X,\Delta Y$) sont calculés à l'avance et stockés dans un tableau en mémoire centrale, le simple transfert vers les registres internes du circuit divise par 2 le débit moyen des vecteurs,
- si l'on connaît ΔX et ΔY en valeurs signées (calculées par différence des coordonnées extrémités-origines), il faut prendre la valeur absolue et envoyer les signes séparément lors de la commande de lancement du vecteur,
- l'implémentation du générateur de vecteurs est sur 8 bits, soit des vecteurs de 0 à 255 pixels. Un vecteur de 512 pixels doit être tracé en 3 fois!

De toutes façons, quelle que soit la capacité du générateur de vecteurs, il faut prévoir:

- une routine de découpage des vecteurs (par dichotomie par exemple) pour que leur longueur n'excède pas celle du générateur,
- une routine de clipping (car le dispositif incorporé peut toujours être insuffisant).

Tout ceci prend du temps et ramène le temps moyen de tracé d'un vecteur à ce qu'il serait en programmé. En effet, si l'on fait le tableau ci-dessous, on obtient les courbes de la figure 2.11 que l'on peut interpréter en pensant que les vecteurs courts sont de loin les plus fréquents (penser au tracé de courbes).

opération	câblé (9365)	programmé (6800)	programmé (68000)
découpage longueur	~ 100 μs	0	0
clipping	~ 100 μs	~ 100μs	~ 10 μs
initialisation	~ 50 μs	0	0
tracé (par pixel)	~ 1 μs	~ 15 μs	~ 1,5 μs

En fait, il est prévu dans le circuit d'exécuter des vecteurs courts en une seule commande (sans charger les registres ΔX et ΔY), mais ceci est inutilisable puisque la routine de préparation des codes de ces vecteurs prend davantage de temps que leur tracé par le circuit.

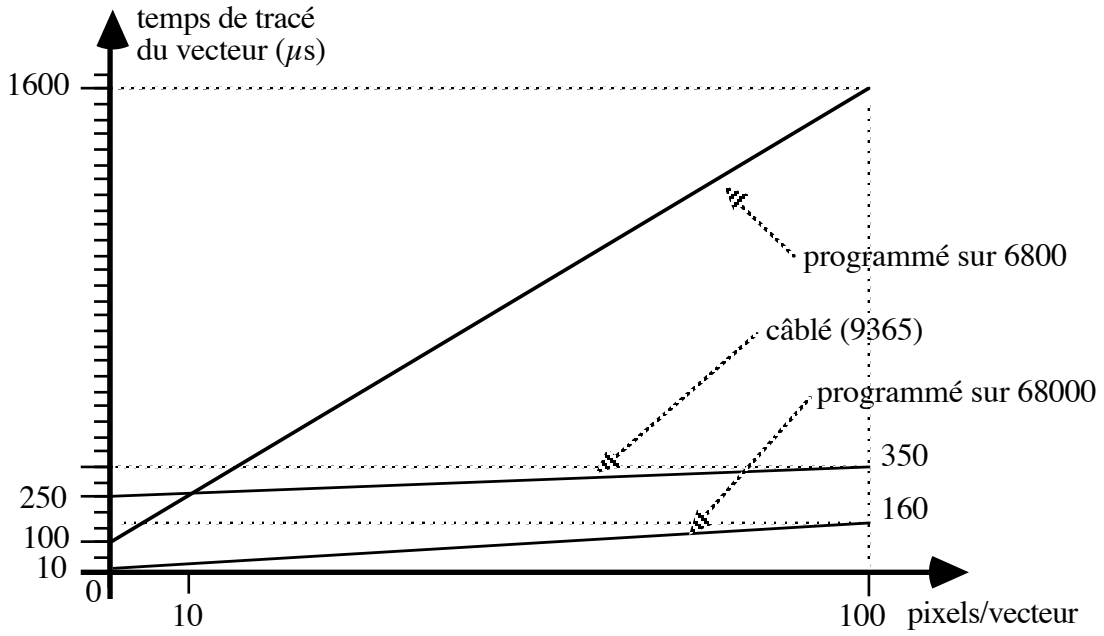

Fig. 2.11: Comparaison de performances

Cette discussion a été conduite avec en tête l'utilisation d'un microprocesseur 6800. Avec un 68000, le temps de tracé par pixel en programmé devient de l'ordre de ce qu'il est en câblé. On gagne en outre les temps de découpage des vecteurs et d'initialisation des registres.

On peut imaginer accélérer le processeur graphique par l'utilisation d'une technologie plus moderne et descendre ainsi la droite correspondante sur le graphique. Le gain ne sera pas très grand car la limitation de ce circuit est liée au temps de cycle des mémoires, puisqu'il écrit un pixel par cycle, et que le temps de cycle des mémoires ne s'est pas abaissé aussi vite que celui des processeurs dans ces dernières années.

Si l'on cherche à écrire plusieurs pixels par cycle mémoire (en utilisant plusieurs générateurs de vecteurs en parallèle par exemple), on optimise notablement le tracé des vecteurs horizontaux ou presque horizontaux, mais uniquement de ceux-ci, ce qui en moyenne sur des vecteurs de pente aléatoire n'est pas très rentable.

Le calcul précédent est fait en considérant que le programme graphique passe tout son temps à tracer des vecteurs dont les coordonnées absolues sont déjà prêtes en mémoire. Dans un programme réel, de CAO par exemple, si l'on compte les calculs et les accès disques, on s'aperçoit que le programme passe moins de 5% de son temps à tracer des vecteurs, et qu'un gain même très élevé sur cette fraction n'apporte rien de tangible.

Notons que ces arguments sont de 2 types:

- une prise en compte de la proportion du temps de tracé des vecteurs dans un programme réel,
- une étude statistique des tracés de vecteurs où l'on s'aperçoit qu'on trace souvent des vecteurs courts.

Lien avec le BitBlT

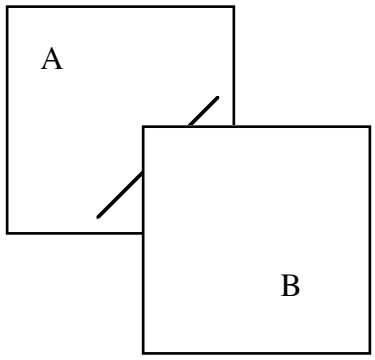

Fig. 2.12: Vecteur et gestion de fenêtres

Si l'on ajoute une gestion de fenêtres, le vecteur est loin d'être une primitive car le clipping sur la fenêtre oblige à intervenir dans le calcul d'erreur: dans la figure 2.12, si l'on enlève la fenêtre B, on doit retracer la partie précédemment cachée du vecteur, et pour que celui-ci se raccorde correctement, sa génération doit prendre en compte l'état précis de la variable s de l'algorithme de Bresenham. Si le générateur de vecteur est câblé, il faut pouvoir intervenir sur cette variable. La complexité ainsi ajoutée à la circuiterie ne peut que ralentir davantage celle-ci.

Conclusion sur ce circuit

L'intérêt réel du circuit EF9365, à l'époque de sa sortie commerciale, a résidé dans 2 aspects:

- les microprocesseurs ne pouvaient pas adresser la mémoire graphique,
- les vitesses des microprocesseurs, des mémoires, des bus, étaient insuffisantes pour partager la mémoire entre la visualisation et un bus modulaire de l'époque.

Ces problèmes ont été levés avec l'apparition des microprocesseurs 16 et 32 bits comme le 68000, et nous verrons dans le chapitre suivant que cela a directement débouché sur le Blit et le Macintosh, qui datent des années 80.

La vie de cette architecture de co-processeurs graphiques n'aurait donc du durer que pendant la période 1978-1983 environ !

A l'heure actuelle, toutes les stations de travail utilisent largement les fenêtres, les pop-up menus, les icônes, etc. La primitive importante est donc le BitBlT. Il se peut que dans une fenêtre on cherche à tracer rapidement des vecteurs. On doit avoir une conception descendante de ces problèmes. Commencer par câbler une primitive pour s'apercevoir ensuite qu'elle n'accélère que 5% des cas est inutile. Cette constatation est de même nature que celle qui a conduit aux processeurs RISC. Il est indispensable de concevoir en même temps l'arborescence du logiciel et le matériel. A l'heure actuelle, le seul matériel qui peut accélérer ces fonctions est un processeur général classique plus rapide.

En fait, après avoir vu les limites d'une telle architecture spécialisée, on en conclut que la notion de "primitive-vecteur" n'était liée qu'à la structure "terminal-relié-par-une-ligne-lente". C'est la condensation de l'information sur la ligne qui était importante, et non le fait qu'un type d'objet soit dessiné souvent.

Les co-processeurs, en câblant quelques primitives, créent un parallélisme avec 2 processeurs de complexités et de technologies comparables, et posent donc les problèmes classiques de synchronisation, alors que l'évolution de la technologie gagne ce facteur 2 presque chaque année, et que le besoin réel en puissance est 100 à 1000 fois supérieur (par exemple pour rendre la synthèse d'image 3D interactive!).

Les autres co-processeurs graphiques

Le EF9365 (sorti en 1980) se caractérise par:

- une prise en charge des adresses de la mémoire graphique uniquement, pas des chemins de données,
- peu de formats d'affichage différents, seulement ceux possibles sur un balayage télévision 625 lignes. (Ceci est lié à la recherche de faible coût du moniteur. On peut se demander pourquoi économiser sur le contrôleur graphique si celui-ci oblige à acheter un moniteur à 20 kF!).
- 2 primitives seulement: vecteurs et caractères.

Le circuit sorti ensuite a été le NEC 7220, complètement débogué vers 1983, qui comprend en outre [OHU*81]:

- format d'affichage programmable (jusqu'à 1000 x 1000), avec zoom et scroll,
- prise en charge du traitement des données jusqu'à 4 bits par pixel,
- remplissage de polygones,
- FIFO de commandes du côté μP.

Depuis peu, une nouvelle génération de co-processeurs a vu le jour:

- AMD QPDM
- NS AGCS [CB86]
- NCR 7300 et 7301
- Intel 82786 [Shi86]
- Texas-Instruments TMS34010 [ASP*86]
- Hitachi ACRTC

dont les principales caractéristiques sont signalées dans [Con87]. Fondamentalement, ces circuits sont peu différents de ceux de la génération précédente. Ils sont essentiellement plus rapides car profitent de technologies plus modernes, mais le gain sur le temps d'accès des mémoires n'est pas suffisant pour que les choses soient très différentes.

Chapitre 3: Architectures de machines monochromes orientées BitBlT: Alto, Dorado, Macintosh, Blit, Sun, Apollo

Présentation

Il peut paraître curieux de décrire ici des ordinateurs généraux (Alto, Dorado, Macintosh), alors que notre propos est de parler de contrôleurs d'écrans, mais ces machines ont deux particularités (contradictoires sans une étude plus approfondie):

- ce sont les machines les plus performantes pour l'exécution de la fonction BitBlT,
- elles possèdent peu de circuiterie spécialisée pour le graphique.

En fait, il ne s'agit pas de machines pour lesquelles la sortie graphique est un périphérique rajouté sur une machine générale, mais de machines conçues dès le départ pour obtenir de bonnes performances graphiques.

La première est l'Alto, sur laquelle a été inventé le BitBlT. Les autres en sont directement issues, chacune apportant une fonctionnalité nouvelle: le Dorado est la recherche de performances supplémentaires, le Macintosh est la recherche d'un coût minimum, le Blit est un terminal, même s'il contient un processeur puissant et une mémoire importante, les Sun et Apollo s'écartent un peu de cette lignée dans la mesure où ils contiennent davantage de circuiterie spécialisée et où ils traitent la couleur.

L'Alto

Cette machine a été développée au laboratoire de Xerox PARC dans la période 1973-1975 [TML*82]. C'est sur cette machine qu'ont été mis au point les concepts graphiques de Smalltalk qui ont débouché sur l'utilisation très générale du BitBlT comme primitive graphique [Ing81]. C'est aussi sur cette machine qu'a été développé le premier réseau Ethernet (alors à 3 M bits/s).

Fig. 3.1: L'Alto, d'après [TML*82]

C'est à la fois le premier ordinateur individuel et l'ancêtre des stations de travail (voir figure 3.1).

A l'origine de ce projet se trouve l'envie d'Alan Kay de réaliser le "Dynabook" ou "Livre Dynamique", ordinateur individuel de la taille d'un livre, permettant de manipuler sur un écran des documents tels que: pages imprimées, illustrations, objets de bureau, etc. C'est aussi le concept à la base de la "bureautique" [Kay77]. Ce projet du début des années 1970 n'est que tout juste réalisable aujourd'hui, aussi l'Alto devait être une première étude de faisabilité, avec la technologie de l'époque. Les nouvelles idées liées à ce projet sont: écran "bit-map", fenêtres, icônes, utilisation de la souris, Ethernet, serveurs fichiers et imprimantes...

Le projet a débuté en 1973, peu après la sortie du premier microprocesseur, le 4004 de Intel (1971). Ce circuit n'était pas d'une puissance compatible avec le projet Alto. Ce n'est que vers 1978 que sont apparus les ordinateurs individuels à base de microprocesseurs (8080, 6800, 6502,...).

La technique utilisée pour l'Alto a donc été de construire une machine microprogrammée (cycle 170 ns) à partir de circuits MSI. La partie opérative utilise l'UAL 74181 et les registres 74189. Les premières versions comprennent 1 K micro-instructions en PROM, les suivantes 4 K dont 3 en RAM.

L'originalité de cette machine réside dans la gestion des entrées - sorties. En effet, elle inclut 3 périphériques rapides: Ethernet (0,4 M oct/s), disque (1 M oct/s), et surtout écran "bit-map" (2 M oct/s). Ces débits ne se rencontraient pas à l'époque dans une petite machine. Il a été nécessaire d'associer les entrées - sorties de façon très étroite au coeur de la machine (voir figure 3.2):

- le processeur d'E/S est l'unité centrale elle-même, avec ses registres,
- le chemin de données de ces E/S est connecté directement aux bus de l'UAL,
- le séquencement des E/S est réalisé directement par microprogramme.

Cela aboutit à un partage de l'unité centrale qui est commutée entre des micro-tâches, au nombre de 16, à priorités fixes. L'une est l'émulation du code machine (BCPL, un ancêtre de C), les autres sont réparties entre les E/S (3 pour la visualisation). Pour que la commutation de tâches ne consomme aucun temps, le registre d'adresse de micro-instruction est répliqué 16 fois (RAM de 16 mots) et un encodeur de priorité observant les requêtes permet de changer de tâche sur un front de l'horloge.

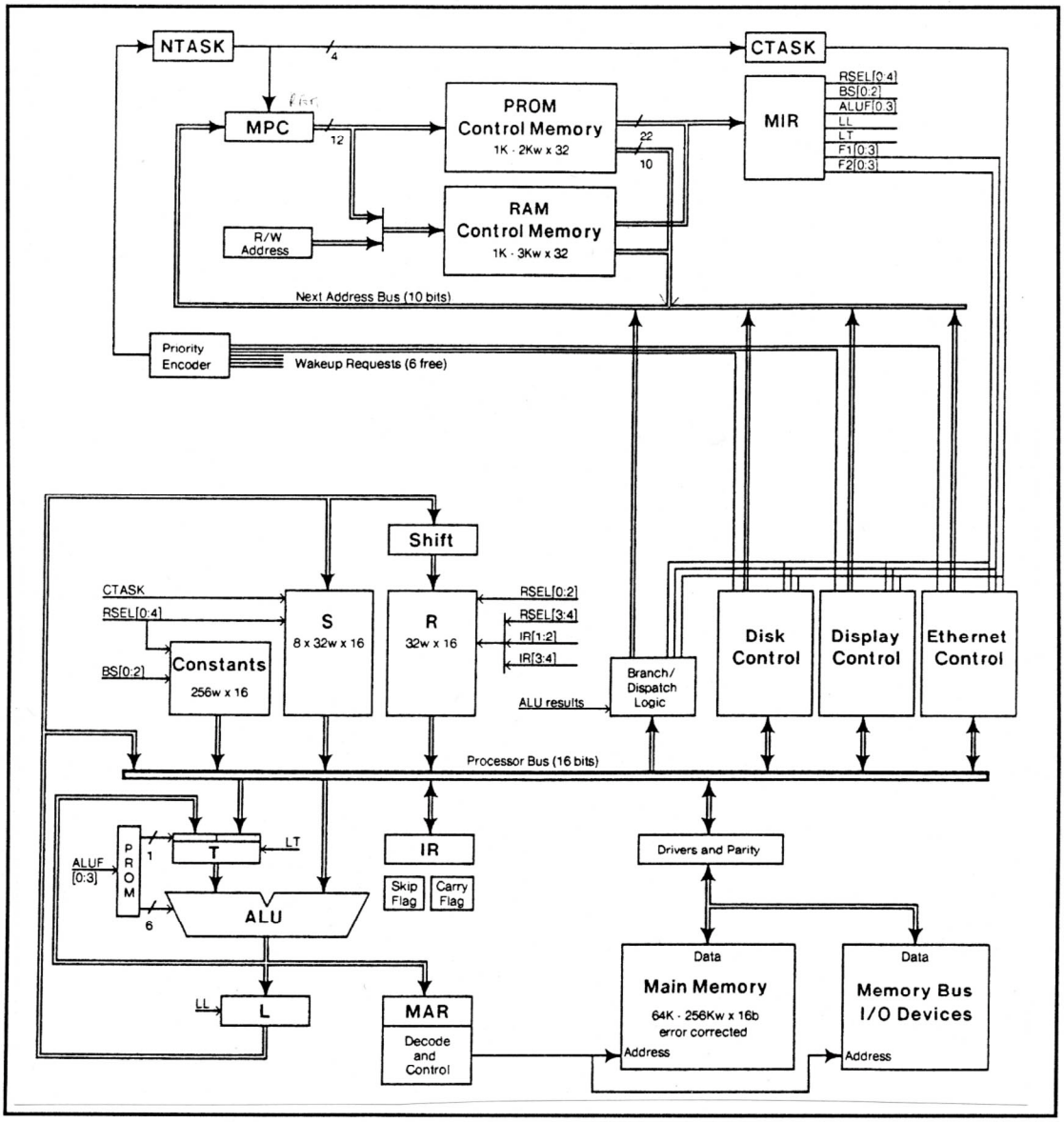

Fig. 3.2: Le processeur de l'Alto, d'après [TML*82]

Ceci permet de simplifier au maximum les coupleurs d'E/S qui bénéficient chacun d'une "tranche" du processeur principal et de toutes les ressources de l'émulateur programme (arithmétique, registres, gestion mémoire, procédures microprogrammées). Il est par exemple inutile de rajouter un DMA qui peut être avantageusement remplacé par une micro-tâche. Le débit possible d'échange avec la mémoire par cette méthode est 4 M oct/s (2 mots de 16 bits en 6 micro-cycles).

La mémoire ne voit donc qu'un interlocuteur: l'unité centrale. Il n'y a pas de bus mémoire et d'allocateur complexe pour partager celle-ci entre les E/S. C'est le partage de

l'unité centrale qui induit le partage de la mémoire. Il est à noter que la mémoire et le balayage de l'écran sont synchrones avec l'unité centrale. Tout ceci conduit à une structure mémoire extrêmement simple, avec un débit proche des limites technologiques.

Structure du coupleur écran

L'écran est rectangulaire, orienté verticalement (portrait), orientation induite par l'envie de manipuler principalement des pages de texte. La résolution est 808 lignes de 608 points. Le balayage comprend 875 lignes rafraîchies à 30 Hz, en 2 trames entrelacées (60 Hz). Le clignotement est compensé en partie par le choix du phosphore lent P40.

Le débit moyen est donc de 808 x 608 x 30 = 14,73792 MHz (2 M oct/s).

Le débit en pointe (hors des marges) est 20 M bits/s soit 50 ns par pixel (exactement 53 ns).

L'image à afficher est rangée en mémoire centrale de la façon suivante: on définit un "bit-map" comme un rectangle de h lignes de w points (w multiple de 16, largeur du mot mémoire). Un tel "bit-map" est rangé en mémoire de façon séquentielle: les 16 bits consécutifs d'un mot, puis les mots d'une ligne à la suite dans l'ordre des x croissants puis les lignes consécutivement par ordre des y croissants (axe orienté vers le bas de l'écran). Un "bit-map" est donc une structure de données, pas forcément liée à l'écran. On crée des structures chaînées de "bit-map". Une fenêtre sur l'écran correspond à un "bit-map".

L'écran physique est un "bit-map" de dimension particulière, situé n'importe où en mémoire. Le coupleur visu connaît un pointeur désignant l'adresse de base du "bit-map" à afficher. Le BitBlT est alors défini comme la fonction de combinaison généralisée de "bit-map": copie, clipping, opération pixel à pixel.Le chemin de données du coupleur visu (voir figure 3.3) est constitué simplement d'un FIFO suivi d'un registre à décalage, le tout de largeur 16 bits. Le registre à décalage permet de passer d'un cycle de 50 ns par pixel à un cycle de 16 x 50 = 800 ns par mot. Le FIFO sert à compenser les irrégularités de l'allocation du processeur à la tâche de visualisation.

Des compteurs spéciaux de la carte visu génèrent le séquencement des lignes et des trames pour réveiller les tâches et fabriquer le signal de synchronisation pour le moniteur. Trois tâches sont nécessaires:

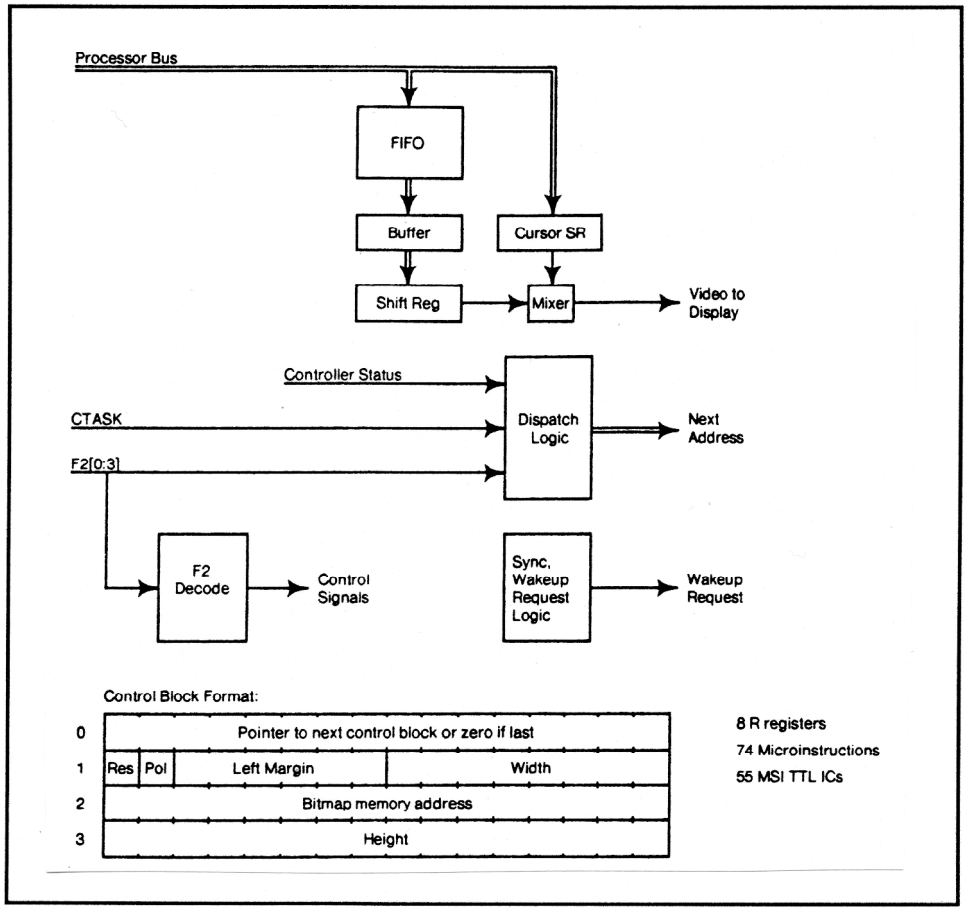

Fig. 3.3: Le contrôleur graphique de l'Alto, d'après [TML*82]

- "Data Task": transfère les données de la mémoire vers le FIFO, et met des 0 lors des marges. Consomme les 2/3 de l'unité centrale car "fetche" les mots de 32 bits en 1μs qui sont visualisés en 1,6μs,

- "Line Task": une fois par ligne vidéo, soit toutes les 38 μs, initialise la largeur de marge, l'adresse du "bit-map" et sa largeur,

- "Field Task": une fois par trame vidéo.

Le curseur (matrice 16 x 16) est ajouté au niveau de la vidéo. Sa gestion occupe une tâche supplémentaire.

Ecriture sur l'écran

Toutes les opérations d'écriture sur l'écran sont effectuées par des écritures en mémoire générale, le plus souvent par appel de la fonction BitBlT microprogrammée. L'écriture de caractères alphanumériques à partir de fontes chargées en mémoire est effectuée par une fonction voisine mais optimisée pour la taille de rectangle correspondant aux caractères.

Réalisation de la machine

Le processeur est implémenté en 5 cartes de 70 circuits MSI chacune. Chacune des trois E/S rapides consiste en 1 carte de 60 circuits. La mémoire (mots de 16 bits avec correction d'erreur) comporte 312 circuits soit 256 K mots dans la version avec des circuits de 16 K x 1 bits.

Conclusions

Une des conclusions des concepteurs de cette machine est que la puissance des E/S est de plus d'importance que celle de l'émulateur du code programme. A l'époque, aucune autre machine ne possédait d'écran "bit-map" et de réseau Ethernet. Les fonctionnalités apportées ainsi (préparation interactive de documents pour un serveur imprimante laser, illustration graphique des documents, partage d'un serveur fichier, courrier électronique) ont montré l'intérêt d'un réseau d'ordinateurs individuels.

Ce que l'on aimerait rajouter à cette machine et qui a conduit au Dorado, est d'une part de la mémoire et d'autre part de la puissance pour l'unité centrale. Mais ce qui a été mis en évidence par l'Alto est que le partage de l'unité centrale n'est pas un inconvénient dans la mesure où le facteur limitant est le débit global de la mémoire, et que celui-ci peut être plus élevé si la mémoire n'est pas partagée. En outre, si l'on ajoute un cache, comme dans le Dorado, il en suffit d'un seul avec cette architecture.

Du point de vue de la gestion du BitBlT, il est intéressant de constater l'absence de circuiterie spécialisée. Aujourd'hui, pour obtenir de bonnes performances sur le BitBlT, on est quelquefois amené à construire des circuiteries très complexes. Et pourtant, ce sont les performances du BitBlT sur l'Alto qui ont convaincu de l'intérêt de cette primitive!

Cette apparente contradiction est liée au fait qu'on cherche à l'implémenter sur des architectures radicalement différentes, basées sur un microprocesseur, où c'est la mémoire qui

est partagée entre les E/S par un bus modulaire. Ce partage ralentit le débit total de la mémoire, ce qui induit souvent la spécialisation d'une mémoire graphique.

Le Dorado

L'expérience de l'Alto a montré que cette architecture était très bien adaptée à la manipulation d'images complexes mais que la machine péchait par insuffisance de puissance, la visualisation consommant les 2/3 de la puissance de l'unité centrale. L'équipe CSL à Xerox Parc a donc entrepris (1975-1979) le développement d'une machine beaucoup plus performante mais en gardant cette architecture avec une unité centrale partagée entre les E/S. Le débit mémoire a été porté à plus de 60 M oct/s. Un cache a été intercalé entre l'unité centrale et la mémoire et une "Instruction Fetch Unit" (IFU) a été mise en parallèle avec l'unité centrale (voir figure 3.4). La taille mémoire peut aller jusqu'à 8 M octets.

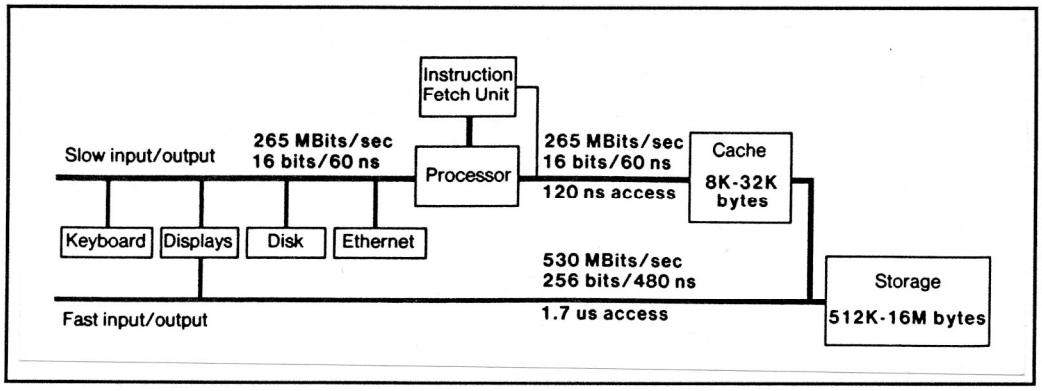

Fig. 3.4: Architecture du Dorado, d'après [Pie83]

Le temps de cycle micro-instruction est de 60 ns et il y a 4 couches de pipe-line (voir figure 3.5). La technologie est l'ECL 10 K. L'IFU est capable de préparer les instructions à une fréquence moyenne proche de 16 MHz, soit la fréquence des micro-instructions. La plupart des instructions sont exécutées en une micro-instruction. La probabilité que l'adresse demandée soit dans le cache est en pratique de 99%. Tout ceci donne à cet ordinateur individuel une puissance de "mainframe" de l'époque [LP80] [CLP81] [Pie83].

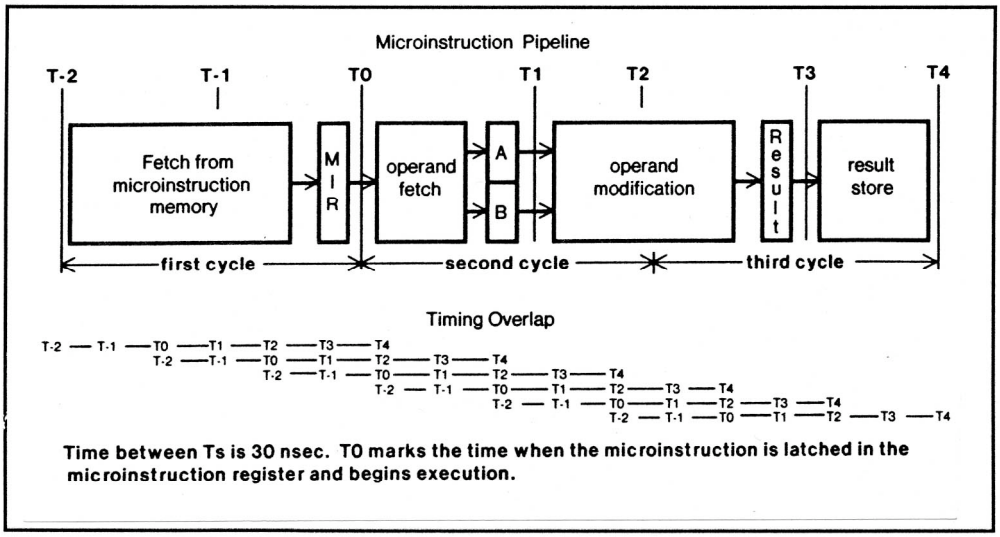

Fig. 3.5: Pipe-line du Dorado, d'après [Pie83]

Le partage de l'unité centrale a été systématisé. Alors que dans l'Alto seul le registre d'adresse microprogramme était répliqué en 16 exemplaires, ici les registres et les piles de travail le sont aussi. Le numéro de tâche en cours doit être communiqué dans toute la machine, vu la structure très pipe-linée.

Les bus et l'UAL sont sur 16 bits. Il est inclus un décaleur-masqueur à l'entrée de l'UAL pour les opérations de BitBlT microprogrammé (voir figure 3.6).

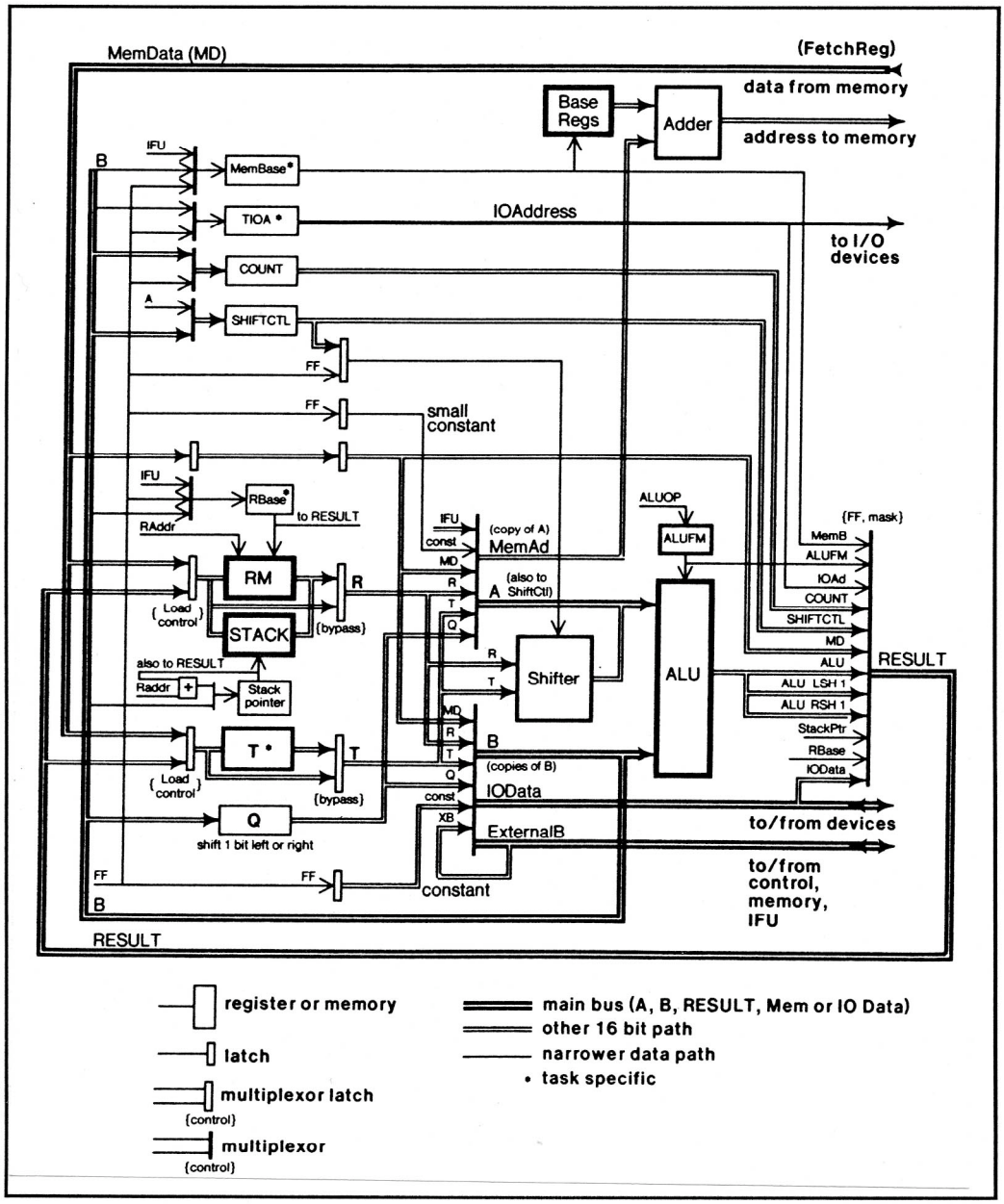

Fig. 3.6: Processeur du Dorado, d'après [Pie83]

Deux types de visus ont été installés:

- un écran monochrome 1024 x 808 x 1 bits orientation paysage (8% de la bande passante mémoire)
- un écran couleur 640 x 480 x 8 bits (40% de la bande passante mémoire).

La visualisation demande uniquement 2 micro-instructions pour transférer un bloc de données. Le BitBlT est le plus efficace qui ait jamais été réalisé.

La machine est utilisée comme station de travail individuelle bien qu'elle soit trop volumineuse et consommatrice pour être située dans un bureau (plus de 2000 W ! - voir figure 3.7).

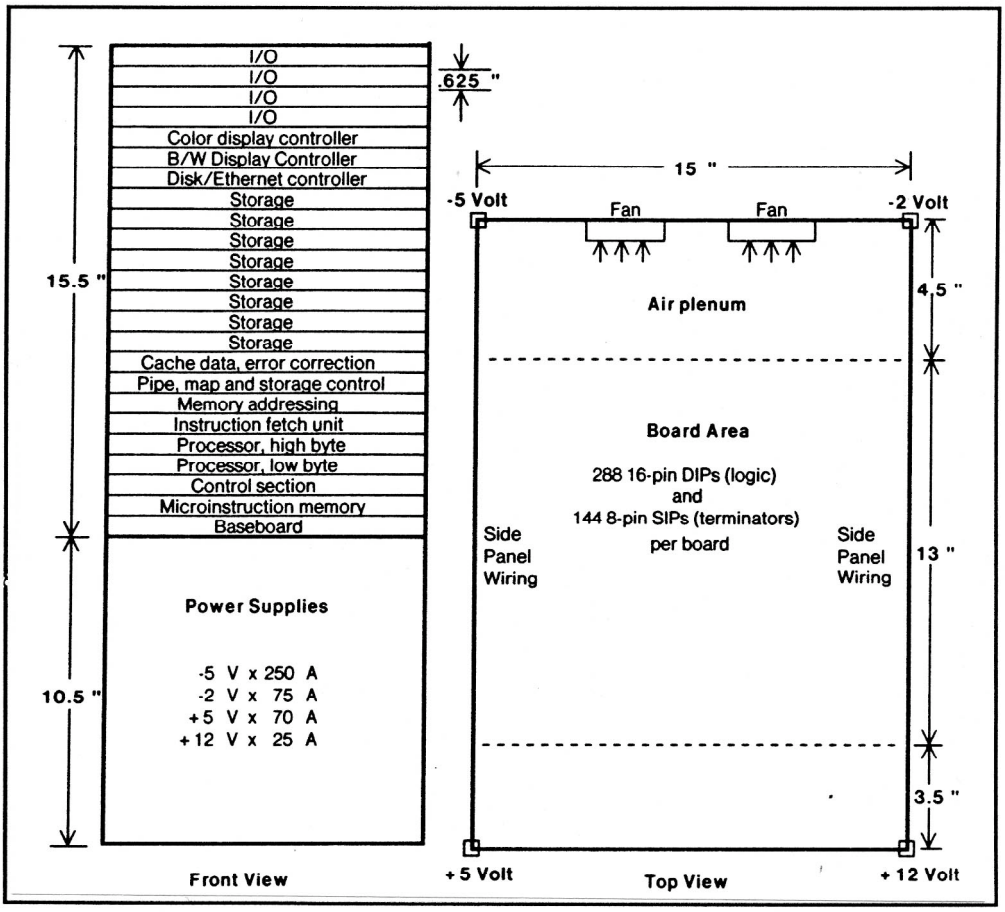

Fig. 3.7: Châssis du Dorado, d'après [Pie83]

Le Macintosh

L'idée d'Apple en créant cette machine (développée de 1981 à 1983) a été de rendre commerciales les utilisations d'un écran monochrome développées à Xerox Parc. Outre les fonctionnalités ont été gardées les idées suivantes:

- mémoire d'image en mémoire générale (occupant environ 20 K octets sur 128 K),
- BitBlT programmé.

Seulement, ceci n'est possible qu'aux conditions suivantes:

- débit mémoire suffisant,
- puissance de l'unité centrale suffisante,
- capacité d'adressage par l'unité centrale suffisante.

Les 2 derniers points sont résolus grâce au microprocesseur Motorola 68000 [Wil84]. Mais le premier est très critique, surtout si l'on souhaite réaliser une petite machine bon marché.

Sur Alto et Dorado, le débit mémoire élevé est rendu possible par le fait que la mémoire n'est pas partagée entre les E/S mais n'est vue que par l'unité centrale qui est elle-même partagée. Ceci n'est pas transposable à une unité centrale mono-circuit commerciale comme le 68000. Si l'on souhaite partager la mémoire grâce à un bus modulaire comme dans la plupart des machines à base de μP, on est limité à des débits mémoire assez faibles à cause des temps d'allocation et des problèmes de synchronisation, surtout si l'on s'impose un faible coût.

La solution choisie a été d'une part d'abandonner la modularité du partage de la mémoire, en considérant que sont connues dès la conception toutes les E/S possibles, et d'autre part de choisir pour les E/S et l'unité centrale une horloge synchrone avec l'horloge des cycles mémoire. Ceci permet une pré-allocation de tous les cycles mémoire dès la conception de la machine. Une base de temps unique sert pour tous les séquencements des différents organes (voir figure 3.9):

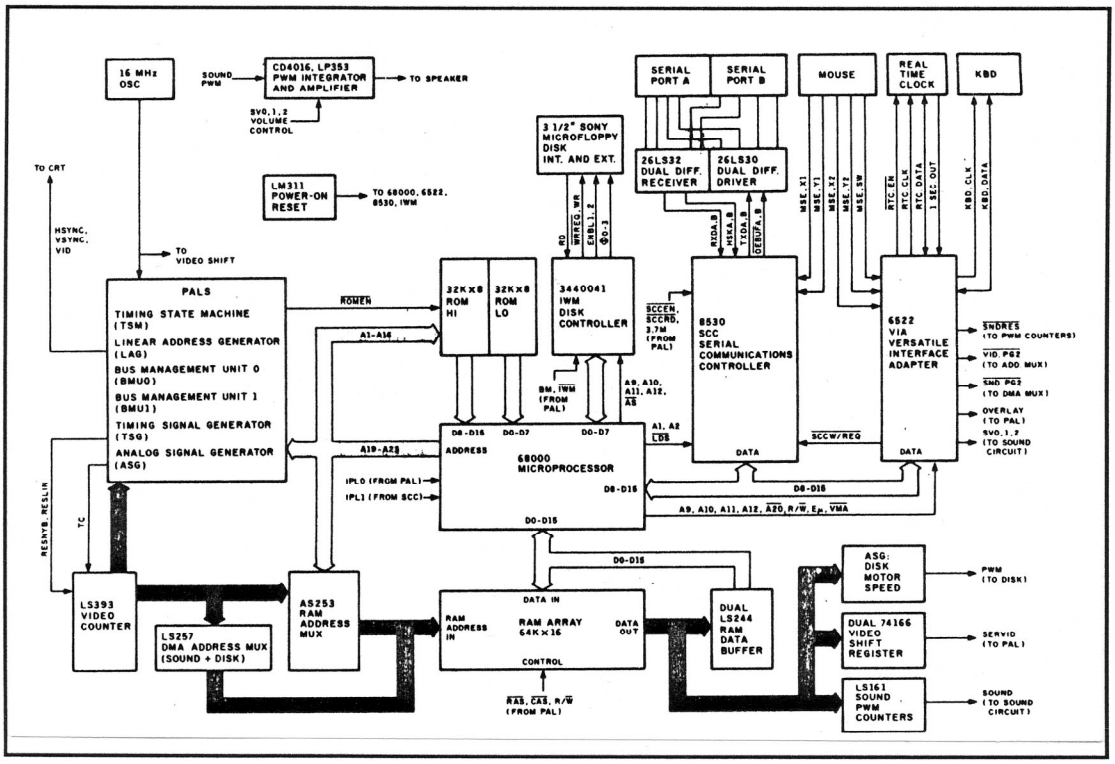

Fig. 3.9: Schéma du Macintosh, d'après [Wil84]

- base de temps de la visualisation, génération du signal de synchronisation du moniteur vidéo,
- base de temps de la sortie sonore,
- disque, port série, clavier, souris.

Tout ceci est fait par un compteur qui sert en même temps de générateur d'adresse pour le DMA de visualisation.

Les concepteurs ont appelé cette technique: "Preallocated Synchronous Time Slots" en comparant ces "slots temporels" aux "slots physiques" d'extensions périphériques des autres machines. En fait, cette idée est la transposition directe des micro-tâches de l'Alto.

L'ensemble conduit à une machine très simple: 50 circuits standards, soit moins que, seul, le coupleur visu ou disque d'autres machines!

Détails du séquencement

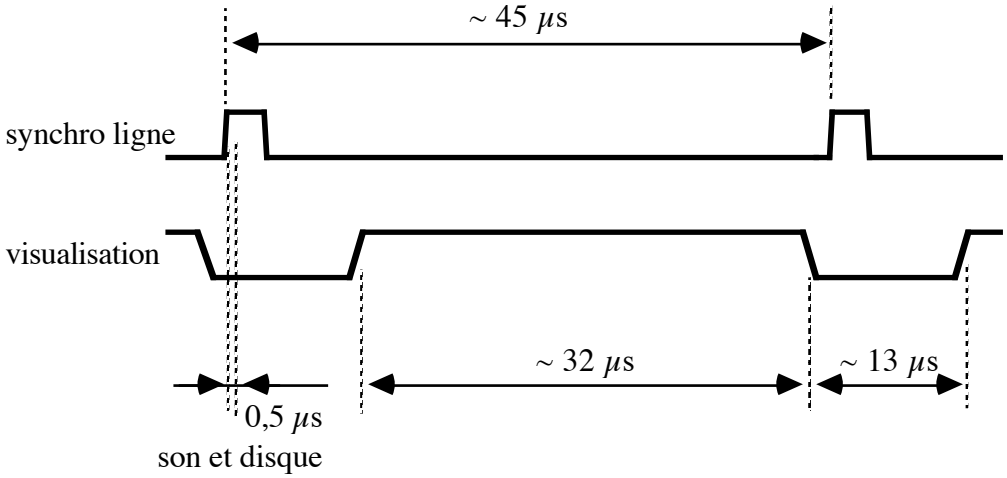

Fig. 3.10: Pré-allocation des cycles mémoire du Macintosh

L'écran a une résolution de 512 x 342, balayé à 60 Hz, la fréquence pixels est 15,67 MHz (la fréquence du microprocesseur est la moitié: 7,83 MHz), la fréquence ligne 22,2 kHz. Comme la machine est organisée en mots de 16 bits, la visualisation consomme un accès mémoire tous les 960 ns. La figure 3.10 décrit le séquencement le long d'une ligne vidéo. Une demi-μs est utilisée en début de chaque ligne vidéo pour la sortie sonore et le changement de la vitesse du disque. Ceci donne donc une base de temps à 22 kHz pour le son.

Le Blit

Cette machine a été conçue dans les laboratoires de Bell pour remplacer les terminaux alphanumériques traditionnels (24 lignes de 80 caractères style VT100 ou VT220) par des écrans "bit-map" profitant des avantages de la primitive BitBlT: fontes alphanumériques variées et de qualité, fenêtres permettant de simuler plusieurs terminaux sur un seul écran, pop-up menus, icônes, interactivité avec une souris [Pik84]. Cette étude a débouché sur un produit commercial: le Télétype DMD5620.

Ce terminal devant être connecté à un hôte sous UNIX via une ligne RS232 de débit moyen (9600 voire 1200 bauds), il a été choisi de modifier le moins possible le logiciel de l'hôte en utilisant au mieux les fonctions standard d'UNIX pour la multiprogrammation, chaque fenêtre de l'écran correspondant à un terminal virtuel UNIX (voir figure 3.11).

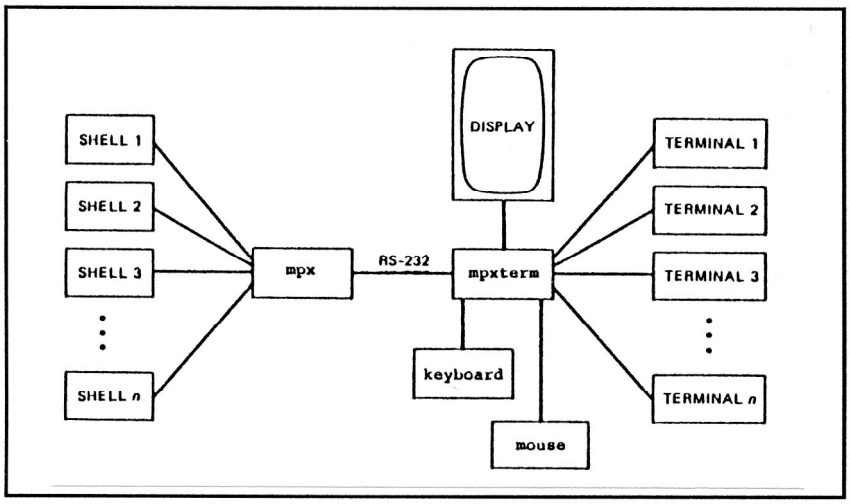

Fig. 3.11: Le Blit, d'après [Pik84]

La ligne RS232 est donc multiplexée entre les n terminaux virtuels utilisant les n fenêtres de l'écran. Il suffit d'ajouter sur l'hôte une procédure de multiplexage des terminaux virtuels sur la ligne, le terminal Blit exécutant une procédure associée pour répartir les sorties sur les fenêtres et décider à quel terminal virtuel correspondent les entrées clavier et souris.

Il a été choisi par les concepteurs (Pike et Locanthi) de reprendre un certain nombre de points des architectures des machines Xerox, mais en utilisant la technologie de 1981:

- écran 800 x 1024 x 1 bits, fréquence image 30 Hz (trames 60 Hz entrelacées),
- mémoire d'image en mémoire générale (256 K en tout, organisée en mots de 32 bits,
- adressage classique unidimensionnel, zone écran fixée uniquement par un registre de base,
- aucune circuiterie spécialisée pour le BitBlT.

Le partage de l'unité centrale n'a pas été repris, en raison du choix du microprocesseur 68000 (à 8 MHz). Mais la mémoire ne voit que 2 interlocuteurs (voir figure 3.12): l'unité centrale et l'écran, les autres périphériques étant connectés à l'unité centrale. Le débit

mémoire pour la visualisation est de l'ordre de 3 M oct/s, soit environ un tiers du débit total de la RAM.

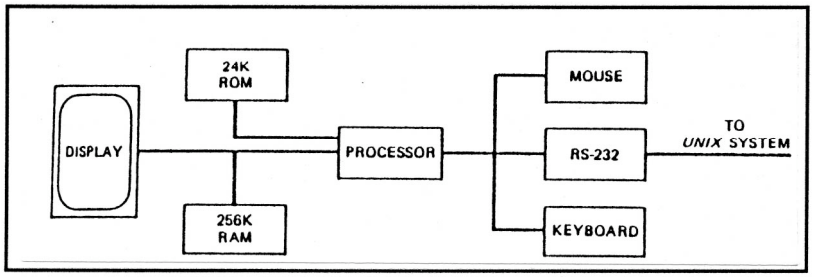

Fig. 3.12: Schéma du Blit, d'après [Pik84]

Cette machine n'a d'un terminal, par rapport à une station de travail, que l'absence de coupleur Ethernet! (beaucoup de stations de travail sont sans disque).

La particularité de cette machine n'est pas tant le matériel que la façon dont est programmé le BitBlT. En effet, les concepteurs, après une étude statistique de l'utilisation du BitBlT ont montré [PLR85] que:

- le cas demandant le plus de vitesse de transfert, et qui est aussi le plus fréquent, est le Scroll vertical. Or ce cas ne demande pas de décalage de pixels puisque l'alignement entre source et destination est le même. Aucune circuiterie spécialisée ne peut donc aller plus vite que le "Move Multiple" du 68000 (pourvu que le mécanisme d'allocation mémoire soit bien conçu).
- le cas fréquent demandant un décalage de pixels est l'écriture de caractères alphanumériques. Or la performance dans ce cas est limitée par l'initialisation des boucles et non par l'efficacité du transfert.
- la zone mémoire correspondant à l'écran est loin d'être la seule sur laquelle peut s'effectuer le BitBlT. Si on ajoute de la circuiterie spécialisée, celle-ci doit pouvoir agir sur toute la mémoire, sinon, il faut traiter 4 types de transferts: écran / écran, écran / hors-écran, hors-écran / écran, et hors-écran / hors-écran.

D'où les conclusions:

- refus d'utiliser une circuiterie spécialisée,

- idée de générer le code du BitBlT pendant l'exécution de la première itération de la boucle externe (transfert de la première ligne du rectangle), pour économiser le temps des tests dans la boucle interne. En effet, une étude de tous les cas de figure du BitBlT montre qu'il n'est pas possible d'écrire une routine pour chaque cas, ceux-ci étant trop nombreux (près de 2000!).

Cette machine permet d'écrire 2500 caractères 9 x 14 par seconde. Le code du BitBlT occupe 1,5 K octets.

SUN 1

Cette station de travail a été développée à l'Université de Berkeley vers 1981 [BBV82], dans le but de multiplier les machines exécutant UNIX, pour les applications de conception VLSI et d'édition en TEX, avec l'idée de récupérer les idées de Xerox Parc, en particulier les performances graphiques et le couplage en réseau local.

Contrairement au Macintosh et au Blit, l'architecture est structurée autour d'un bus modulaire permettant de partager la mémoire entre l'unité centrale et les E/S (Multibus). La conséquence est le relativement faible débit du bus qui conduit à installer une mémoire d'écran séparée pour que le séquencement de la visualisation n'intervienne pas sur celui de la mémoire.

Dans ces conditions, l'efficacité d'un BitBlT programmé s'effondre et il est nécessaire d'ajouter une circuiterie accélérant certaines opérations. Il a été choisi d'ajouter [BB80]:

- un mécanisme d'adressage bidimensionnel permettant de spécifier une adresse (x,y) sur l'écran pour désigner le mot de n bits ($n \leq 16$) constitué des pixels immédiatement à droite du pixel (x,y) (voir figure 3.13),
- adressage possible de plusieurs segments par un jeu de registres (x,y) locaux à la carte mémoire graphique,
- modification d'un mot en 1 cycle par RMW (1 μs).

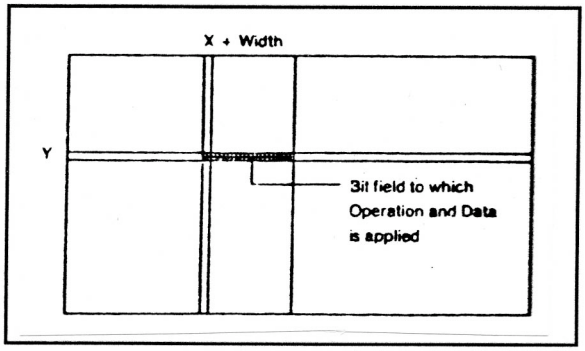

Fig. 3.13: Adressage mémoire sur SUN, d'après [BB80]

Les problèmes de décalage et masquage sont pris en charge par cette circuiterie. Au cours d'un BitBlT, les adresses (x, y) sont fournies par les registres locaux, et la largeur n ainsi que la nature de l'opération sont fournis par l'unité centrale (voir figure 3.14). La taille mémoire graphique est 1024 x 1024, la partie visualisée étant 1024 x 768.

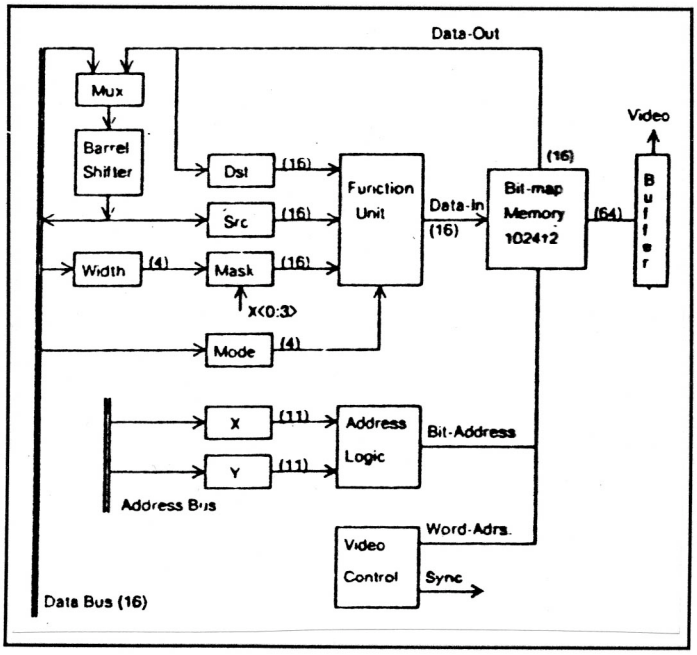

Fig. 3.14: Contrôleur graphique du SUN, d'après [BB80]

Cette architecture est assez efficace et a surtout l'avantage de s'adapter à un bus μP classique. Ses inconvénients majeurs sont:

- la mémoire d'image n'est pas directement adressable par le μP autrement qu'à travers ce mécanisme,

- ce système est 16 bits, ce qui ralentit en fait par rapport à une architecture entièrement 32 bits comme le Blit,

- la mémoire graphique spécialisée, même si elle est un peu plus grande que l'écran (ce qui permet de stocker les fontes alphanumériques), oblige à traiter différemment 4 cas de BitBlT.

SUN 2

Pour cette version suivante, Sun a développé un circuit VLSI spécial intégrant les mêmes fonctions, a créé un bus spécial à haut débit entre unité centrale et mémoire graphique (rendu possible en mettant les deux sur la même carte grâce au gain de place du VLSI), et a rendu la zone mémoire graphique directement adressable par l'unité centrale. En outre, la lecture mémoire pour la visualisation se fait sur une "shadow memory" qui n'est accédée par l'unité centrale que lors des écritures.

Apollo

Cette station de travail, conçue à la même époque, mais commercialisée plus tôt, cherche à résoudre les mêmes problèmes. L'écran 800 x 1024 à 60 Hz est sur une mémoire spéciale, dans l'espace d'adressage du microprocesseur.

Dans une première version (DN400), le BitBlT a été implémenté dans une version réduite (copie de rectangles uniquement) et sur la mémoire graphique uniquement. Dans les versions suivantes, il a été implémenté complètement et sur toute la mémoire.

Circuits VLSI spécialisés

Plusieurs firmes (NS [CB86], Silicon-Compilers, Amd) ont réalisé des circuits d'accélération du BitBlT permettant de construire d'autres machines. Ils reprennent les idées utilisées dans les Sun et Apollo.

Performances comparées de ces machines pour l'exécution du BitBlT

Machine	Vert. Scroll	Horiz. Scroll	8×7 Character	40×40 Texture	Code size, notes
Apollo	25	27	1.0[1]	2.6	hardware (disp. to disp.)
Apollo	642	656			software (disp. to disp.)
Blit	130	370	0.41	1.20	1542 bytes instr's[2]
Dolphin	85.2	128.1	2.02	2.40	~300 micro-instr's
Dorado	23.3	23.8	0.051	0.156	~300 micro-instr's
Ridge	39	112[3]	0.244	0.830	5439 bytes instr's
Sun 1	187	194	1.1	1.89	All separate routines
Sun 2	109	110	0.34	0.82	3344 bytes instr's(disp. to disp.)
Sun 2	82.2	311	0.74	1.78	3068 bytes (mem. to mem.)
Symbolics	30.5	36.8[4]	0.52	1.64	~500 micro-instr's[5]

All times are in milliseconds.
[1] 0.06ms in STORE mode (done in hardware). Scrolling done by hardware in display; much slower if source or destination is not display.
[2] Texturing is a separate operator
[3] 67ms scrolling left. The scrolling tests were 1024wide×800high.
[4] This are memory-memory times. The display memory scrolls about 2.5 times slower.
[5] No clipping, programmer must specify direction of copy

Table 3. Bitblt benchmarks. Thanks to Luca Cardelli, Ken Church, L. Peter Deutsch, Tom Duff, Emden R. Gansner, Peter Langston, George Trow and Dave Ungar for testing the various machines.

Fig. 3.15: Comparaison de performances, d'après [PGI84]

Nous empruntons ici les comparaisons de performances de [PGI84] entre ces machines sur les 4 opérations suivantes: scroll horizontal 800 x 1024 de 1 pixel, idem vertical,

texture 40 x 40 (position aléatoire) et caractères 7 x 8 (position aléatoire). Dans ce dernier cas, si la machine possède une fonction spéciale, celle-ci n'est pas utilisée (voir figure 3.15).

Les remarques que l'on peut faire sont les suivantes:

- la machine la plus puissante (Dorado) ne possède pas de circuiterie spécialisée,
- le Blit est bien placé et utilise pourtant un μP standard 68000 (et pas 68020) à 8 MHz (et pas 16 ou 25).

Ceci est bien expliqué par Pike par ses remarques statistiques. Les circuits spécialisés:

- *optimisent* le cas complexe avec décalage et masquage qui est *rare, et limité par les initialisations*,
- et *ralentissent* le cas simple qui est *fréquent et critique en débit*.

Les cas où les circuits spécialisés sont intéressants (voire indispensables) sont:

- bus modulaire trop lent,
- couleur, car dans ce cas les circuits spécialisés permettent de paralléliser les transferts.

Chapitre 4: Une réalisation personnelle déjà ancienne: carte 512 x 512, 16 couleurs, et sa critique actuelle

Présentation

Ce chapitre décrit la réalisation (ou plutôt résume la suite de réalisations) qui a étayé notre travail sur les architectures graphiques entre 1981 et 1984. Elle est difficilement détachable du contexte de ces années là, contexte lié d'une part au projet de l'équipe dans laquelle ce travail s'insère, et d'autre part à la situation internationale: projets des autres laboratoires, produits commerciaux et leurs évolutions.

La justification de cette réalisation était triple:

- 1) Tout d'abord, il s'agissait de fournir une base d'expérimentation sur les architectures graphiques. On a inclus dans ce but un certain nombre de caractéristiques apportant un maximum de modularité: possibilité d'un nombre quelconque de bits par pixel, possibilité de modifier le traitement vidéo des pixels pour des moniteurs variés, possibilité d'installer cette carte sur des bus microprocesseurs variés, possibilité de gérer la mémoire d'image comme une mémoire spécialisée contrôlée uniquement par le processeur graphique ou bien comme de la mémoire générale de la machine hôte.

- 2) Cette carte était vue comme un prototype, ou une maquette, d'une famille de circuits VLSI spécialisés entrant comme composants d'un micro-ordinateur ou d'une station de travail. Son architecture, c'est-à-dire la répartition des fonctions, la nature des bus de

communication, la fréquence de fonctionnement des différentes parties, était donc liée au découpage en circuits: complexité, nombre de broches, testabilité, généralité.

- 3) Cette carte devait être (et a été) réalisée en 10 exemplaires pour constituer l'outil de travail du Laboratoire. C'est sur ces écrans qu'ont fonctionné les outils de CAO permettant de réaliser le projet de l'équipe. Nous admettions difficilement de travailler sur des outils moins performants que ceux que nous concevions. Cela exigeait une fiabilité suffisante, peu compatible avec un bricolage, mais cela a eu le grand avantage de nous montrer l'impact de la production d'un objet sur sa conception (coût des composants, testabilité, fiabilité), ce qui nous a toujours semblé un point clé pour la définition d'architectures nouvelles.

Après la description des fonctions de cette carte, nous montrons comment une famille de circuits VLSI peut être définie à partir de cette architecture. Le travail de réalisation de ces circuits n'aura finalement pas été entrepris tel quel, mais sur des circuits plus évolués (incluant la fonction BitBlT câblée: projet FLIP / GLOP) par d'autres chercheurs du laboratoire [Dou84] [Kri84] [Par86] [DDPP86]. Entre-temps, nous avons douté de l'intérêt d'architectures graphiques spécialisées, et nous sommes engagés dans la direction opposée, comme le montre cette thèse. Ce chapitre peut donc paraître en opposition avec notre propos, mais il est caractéristique de notre évolution dans la façon de voir les problèmes, et nous sommes persuadés que sans lui, il serait difficile de se convaincre du bien fondé de cette évolution.

Nous incluons donc à partir d'ici des extraits d'un texte ancien [Mat84] (en utilisant une fonte différente), commençant par l'argumentation qui a poussé aux choix d'architecture. Cette argumentation a beaucoup vieilli puisque ces choix ont été faits il y a maintenant près de 6 ans, ce qui est énorme dans ce domaine. La conclusion était d'aller vers davantage de circuiterie spécialisée. Il est intéressant de reprendre les arguments de l'époque comme base de discussion. C'est ce que nous ferons à la fin de ce chapitre, en reprenant la même fonte qu'ici.

Choix de l'architecture globale

Les visus haute résolution couleur rapides, souples, dont nous aimerions disposer sur un ordinateur individuel sont très complexes. De ce fait, ces visus coûtent encore très cher et ne sont pas compatibles avec un ordinateur individuel, mais cela évolue vite et sera bientôt possible. Pour y arriver, il ne suffit pas d'attendre que les architectures actuelles soient intégrables. Le travail est trop important pour ne viser que le but à long terme, et il est nécessaire d'étudier les évolutions des architectures en mettant dans les mains des utilisateurs des produits qui cherchent à réaliser le compromis entre grande puissance et faible coût, avec des composants du moment, ou en étudiant à cette occasion des composants approchant ce but.

Ces produits intermédiaires auront bien sûr une puissance plus faible et un coût plus élevé, mais permettent de cerner les problèmes et de guider les efforts de recherche.

Dans ce cadre, nous avons étudié une visu graphique réalisant un bon compromis puissance / coût pour l'époque présente et le proche avenir, compatible avec un micro-ordinateur d'aujourd'hui, ceci fixant la complexité à 100-150 circuits intégrés.

Cette adaptation du produit au "marché" n'est pas qu'un problème commercial, car les évolutions sont conditionnées par les possibilités techniques. Les écrans couleur pour résolutions élevées restent chers à cause des difficultés de réalisation, et cela doit orienter les choix d'architecture. Le fait que les mémoires les plus denses soient 64 K x 1 bit aujourd'hui (avec un cycle de 300 ns) et bientôt 256 K x 1 bit (200 ns) oriente complètement le degré de parallélisme des opérateurs d'écriture. En outre, nous pensons qu'il faudra, pour les utiliser, que les constructeurs les organisent en 64 K x 4 bits, pour accorder leur débit à celui des visus. Comment, sans faire fonctionner des prototypes, savoir lesquelles de ces contraintes technologiques seront tout à fait dépassées à court terme, et lesquelles ne le seront sûrement pas à très grande échéance ?

Ces considérations nous ont guidées pour fixer les performances d'une réalisation actuelle. Les critères retenus ont été:

- 1) utilisation de mémoires 64 K,
- 2) utilisation d'un moniteur couleur auto-convergent,
- 3) balayage compatible télévision,
- 4) utilisation d'un LSI existant (EF9365) comme contrôleur d'adresses.

Justification de ces choix

Tout d'abord, le dernier point. Un LSI comme le EF9365 remplace environ 300 circuits intégrés. L'utilisation d'un tel circuit est donc indispensable pour le but qu'on s'est fixé. Ce n'est pas une contrainte, car ce circuit ne traite que la gestion des adresses et beaucoup de problèmes restent à étudier dans les chemins de données, la vidéo, les opérateurs pixel, etc..., comme nous allons le voir dans ce chapitre. En outre, il est indispensable de bien cerner les limitations de ces circuits pour concevoir la génération suivante.

Le point 2 est imposé par le prix des moniteurs. Il rejaillit sur la résolution des visus de prix raisonnable aujourd'hui: 500 x 800 maximum en couleur. Il est lié au point 3 par cette résolution ainsi que par l'utilisation à domicile (compatibilité magnétoscope par ex.).

Le point 1 est imposé par la disponibilité dans les années 1981-1983.

Ces contraintes entraînent des limitations de performances qui sont:

- la résolution (point déjà vu).

Nous avons en outre choisi le format "carré" 512 x 512 qui limite le nombre de boîtiers mémoire et la complexité de la périphérie mémoire.

- les modes d'accès à la mémoire limités par le EF9365: le parallélisme entre les boîtiers mémoire ne peut être utilisé que lors de la visualisation. Dans les autres cas, on n'accède qu'à un pixel par cycle, ce qui permet déjà 1 Mpixel/s. En outre, le motif d'accès à la mémoire est rigide: c'est le segment horizontal de 8 bits, ce qui interdit d'effectuer de façon câblée un "scroll" horizontal d'un nombre de pixels quelconque, et interdit un asynchronisme entre vidéo et cycles mémoire.

Ces contraintes n'ont bien sûr pas été acceptées comme inéluctables. Elles ont en fait été choisies comme cadre à l'étude, dans la mesure où elles étaient compatibles avec la réalisation d'une carte intéressante. Plus exactement, elles sont la conclusion de cette étude, dans la mesure où l'analyse qui va suivre n'a pu être faite qu'après avoir pris conscience des problèmes rencontrés en étudiant cette carte. Ces contraintes ont "tout de même" permis les performances suivantes:

- 512 x 512, 16 couleurs dans la version minimum,
- balayage 625 lignes 50 Hz entrelacé,
- carte contrôlée par le circuit EF9365,
- palette de 256 couleurs ou niveaux de gris,

- processeur permettant toute fonction de transcodage de pixel au vol lors de l'écriture,
- "double buffering",
- accès du micro-processeur à la mémoire d'image par segment horizontal ou par pixel,
- entrée caméra vidéo,
- photostyle précis,
- "Roll-up" câblé de l'image graphique.

A ceci ont été ajoutées des fonctions non graphiques, mais essentielles pour les programmes graphiques:

- co-processeur arithmétique en virgule flottante Amd 9511,
- 8 entrées analogiques (pour manches à balai par exemple),
- synchronisation de ou par une source vidéo externe.

Nous allons détailler point par point les originalités importantes de cette réalisation.

Organisation mémoire, parallélisme, double buffering, modularité du nombre de bits par pixel, chemin de données

Nous avons donc choisi un format de 512 lignes et des mémoires 64 K. En se référant aux temps de cycle des boîtiers mémoires, et à leur nombre, nous constatons que nous pouvons réaliser les formats suivants:

- 512 x 512, 8 boîtiers, cycle 320 ns,
- 512 x 640, 10 boîtiers, "
- 512 x 768, 12 boîtiers, "
- 512 x 800, 16 boîtiers, cycle 400 ns.

Nous ne pouvons pas choisir des cycles plus courts avec les mémoires actuelles, car nous avons souhaité, pour les raisons que nous allons voir ci-dessous, effectuer des cycles de "Read-Modify-Write".

Nous avons choisi 512 x 512 pour minimiser le nombre de boîtiers sur la carte. En effet, pour multiplier le nombre de bits par pixel, nous allons devoir multiplier l'ensemble {mémoires + chemin de données} par ce nombre de bits. Le chemin de données n'est pas complexe en nombre de portes, mais proportionnel au nombre de boîtiers mémoire et lourd si on utilise des

circuits standards. C'est une partie qui consommerait peu de surface de Silicium dans un circuit spécial, mais utiliserait beaucoup de broches.

Le choix de 512 x 512 pixels avec 8 boîtiers de 64 K (que nous avons fait pour des raisons de temps de cycle), aboutit à une taille mémoire compatible avec 1024 x 512 pixels. Nous allons utiliser cette capacité pour stocker 2 images et faire du double-buffering. Le changement de buffer se réalisera en commutant le bit de poids fort de l'adresse.

Le choix de ne pas disposer de rotateurs sur la donnée mémoire nous oblige à avoir, comme motif fixe d'accès, le segment horizontal de 8 pixels, dont les adresses se succèdent en séquence le long d'une ligne. Certaines opérations (visualisation, acquisition caméra) sont effectuées sur les pixels en série, à haut débit (8 par cycle mémoire), soit 14 MHz. Ce débit est imposé par le choix du balayage vidéo.

Les modes d'accès à la mémoire sont de 3 sortes, en lecture comme en écriture:

- accès à 8 pixels par cycle, en série à 14 MHz: visualisation, acquisition caméra, effacement, positionnement d'un fond,
- accès à 8 pixels par cycle, en parallèle: échanges avec le bus micro-processeur, 8 pixels sur 1 seul bit (dans un seul "plan"),
- accès à 1 pixel par cycle: écriture d'objets graphiques (vecteurs, caractères, etc...), échanges avec le micro-processeur, pixel par pixel (dans tous les "plans" à la fois).

Tous ces cycles peuvent être en "Read-Modify-Write". L'intérêt est:

- le transcodage au vol des pixels lors du tracé de vecteurs ou de caractères (par l'opérateur "pixel"),
- l'acquisition caméra simultanée avec la visualisation,
- l'effacement d'un buffer simultané avec sa visualisation (indispensable en animation),
- l'utilisation du processeur pixel en transcodage lors d'un accès micro-processeur.

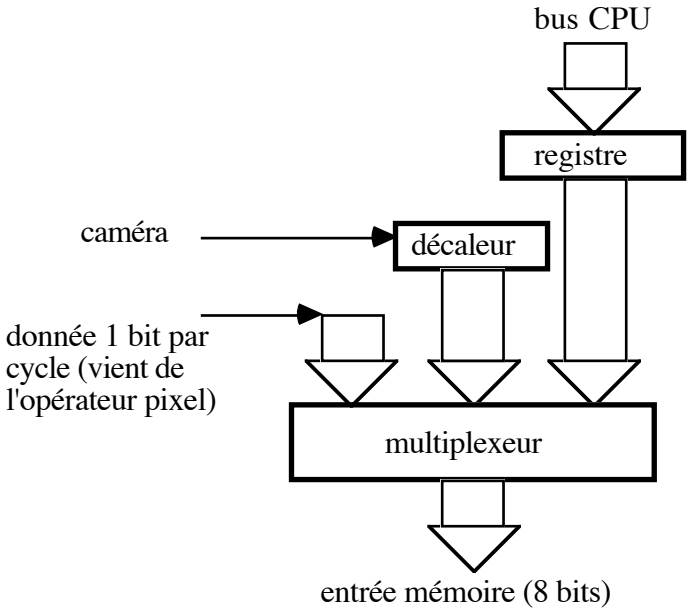

Fig. 4.1: Chemin de données d'entrée mémoire

Ceci débouche sur la structure des chemins de données des figures 4.1 et 4.2. Pour une entrée à un pixel par cycle, le bit de donnée est envoyé aux 8 mémoires. C'est la sélection mémoire, issue des 3 bits faibles de l'adresse, qui assure qu'un seul boîtier est modifié.

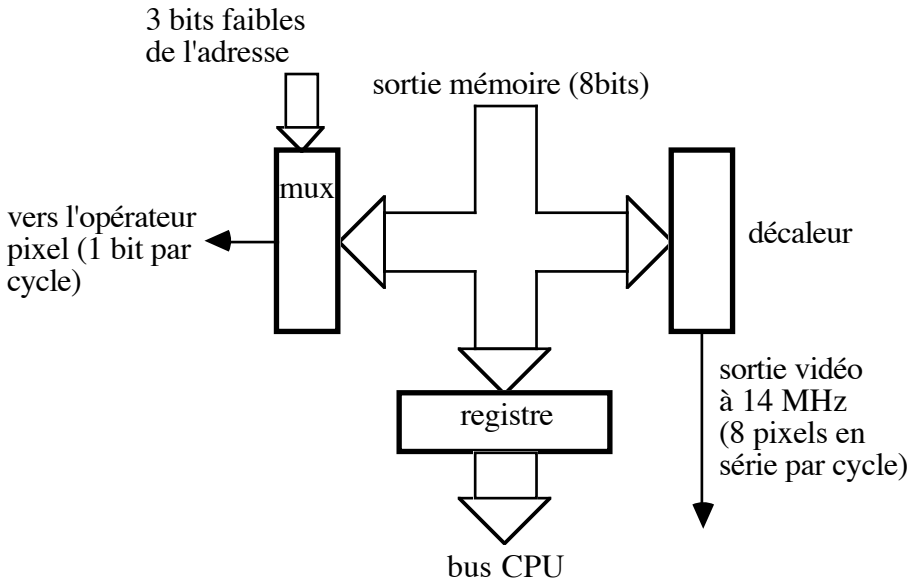

Fig. 4.2: Chemin de données de sortie mémoire

Toute cette partie doit être multipliée par le nombre de bits de chaque pixel, comme nous le verrons dans la figure 4.10.

Le décalage de huit bits à 14 MHz indique un cycle mémoire de 571 ns, soit une fréquence de 1,75 MHz. Nous avons choisi des mémoires rapides permettant de réaliser 2 cycles par 571 ns, dont l'un de RMW. Ceci permet donc d'intercaler systématiquement entre tous les cycles de RMW déjà prévus et utilisés par le EF9365, un cycle de lecture ou d'écriture pour le microprocesseur en "Bit-map". Ceci donne donc une fréquence mémoire moyenne de 3,5 Mcycles, d'où pour 4 plans: 4 x 3,5 = 14 M octets par sec. La capacité est alors de 4 x 8 x 64 K bits = 256 K octets.

Pouvons-nous confondre en un seul les 2 registres à décalage de visualisation et d'acquisition caméra? Pouvons-nous confondre (connecter ensemble) les bus d'échange micro-processeur en lecture et en écriture? C'est possible, mais la volonté d'utiliser largement le RMW nous aurait obligé à rajouter des circuits de registres, pour pouvoir utiliser dans le même cycle et sur le même bus, d'abord les données de lecture, puis des données différentes en écriture. Ceci aurait abouti à la même complexité de bus, et aurait moins bien utilisé le temps disponible dans un cycle mémoire.

Situation de la carte dans l'espace d'adressage du micro-processeur (problème du "bit-map")

Un grand débat pour les visus graphiques est: Situons-nous la mémoire d'image dans l'espace d'adressage du microprocesseur de l'ordinateur hôte? Citons les avantages et les inconvénients:

- avantages: possibilité d'écrire des programmes qui travaillent directement sur les pixels, possibilité d'utiliser un DMA général sur cette mémoire.
- inconvénients: la visualisation va-t-elle consommer de la bande passante de la mémoire ? Cet espace mémoire va-t-il être pris sur ce qui est disponible pour les programmes ? Y aura-t-il d'autres opérateurs de tracé graphique que le microprocesseur ?

En fait, c'est l'argument de place mémoire occupée qui nous a fait choisir puisque le nombre de 4 bits par pixel représente 256 K octets, alors que les micro-ordinateurs 8 bits n'adressent souvent que 64 K octets. Ceci change complètement avec les micro-processeur 16 ou 32 bits. En outre, avec les débits mémoire possibles, l'accès pour la visualisation ne peut que pénaliser l'ensemble de la machine.

Nous avons choisi de permettre un accès direct du micro-processeur s'il a la capacité d'adresser la mémoire graphique (à condition qu'il attende un cycle et se synchronise), et d'intercaler des registres tampons pour le cas où il ne peut adresser directement la mémoire (micro-processeur 8 bits par exemple).

Modularité réelle du nombre de bits par pixels

Si on regarde le nombre de bits par pixel nécessaire suivant les applications, on constate 2 types d'utilisations:

- applications de traitement de texte, développement logiciel, CAO de circuits, etc: alors 16 couleurs suffisent, donc 4 bits par pixel.
- applications de synthèse d'images réalistes ou de traitement d'images: on souhaite beaucoup de couleurs. On utilise souvent 24 bits par pixel.

Nous avons choisi de mettre 4 bits par pixel dans la version de base et de prévoir une extension de la mémoire et de sa périphérie par groupe de 4 bits. Le nombre de circuits intégrés rajoutés pour chaque groupe de 4 bits est d'environ 60.

Palette et DACs

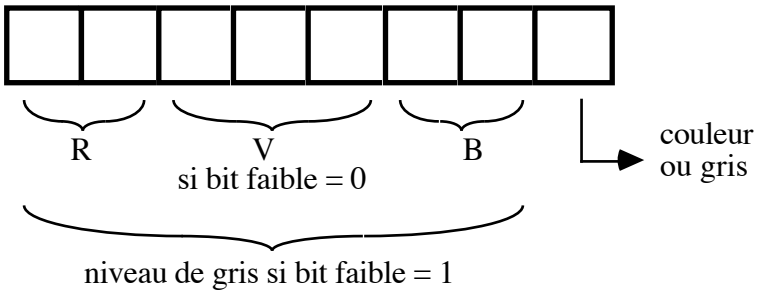

Fig. 4.3: Codage de la palette

Nous avons principalement étudié la palette pour 4 bits par pixel, ce qui est très simple puisque celle-ci ne comporte que 16 adresses. Nous avons choisi de coder la sortie de la palette sur 8 bits, ce qui donne 256 couleurs.

Comme il est intéressant de travailler sur des images en niveaux de gris sur lesquelles on superpose quelques couleurs, nous avons choisi de séparer ces 256 couleurs possibles en 128 gris et 128 couleurs. Le codage est donc celui de la figure 4.3, qui se réalise par le schéma de la figure 4.4.

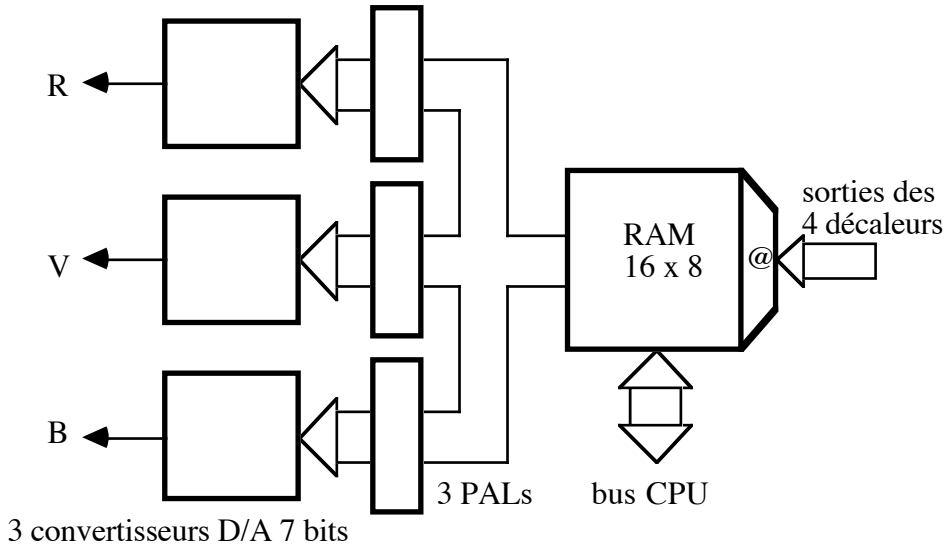

Fig. 4.4: Réalisation de la palette

La RAM est à double accès:

- une adresse vient des registres à décalage. La fréquence des accès est 14 MHz.
- l'autre adresse vient du micro-processeur. Cet accès peut être beaucoup plus lent.

L'allocation est simple: le micro-processeur est prioritaire, mais les programmes ne doivent se permettre de modifier le palette que pendant le retour trame, s'ils veulent éviter de créer des parasites sur l'image.

La logique combinatoire située avant les DACs est, dans notre montage, réalisée par des PAL, circuits programmables. Il est ainsi facile de modifier le transcodage suivant l'application.

Processeur "pixel"

Nous appelons ici "processeur pixel" (ou "opérateur pixel") une circuiterie se chargeant du transcodage du pixel pointé par un processeur de tracé de vecteur ou de caractère. Nous avons indiqué dans les choix de départ que nous ne le parallélisons pas. Comment le choisir? Prendre une fonction purement combinatoire? Ou bien un petit processeur? Vue l'étude portant principalement sur 4 bits par pixel, nous avons choisi de le réaliser par une table de transcodage (RAM de 16 mots de 4 bits). Ceci est facile à réaliser et permet n'importe quelle fonction par modification de la table (figure 4.5). Il est intéressant de comparer ce montage à celui de la figure 4.6. Dans notre montage, l'opérande externe est en quelque sorte contenu dans la RAM de transcodage. La fonction est plus générale, mais plus longue à modifier. Nous avons fait ce choix pour expérimenter toutes sortes de fonctions et retenir ultérieurement les plus utiles.

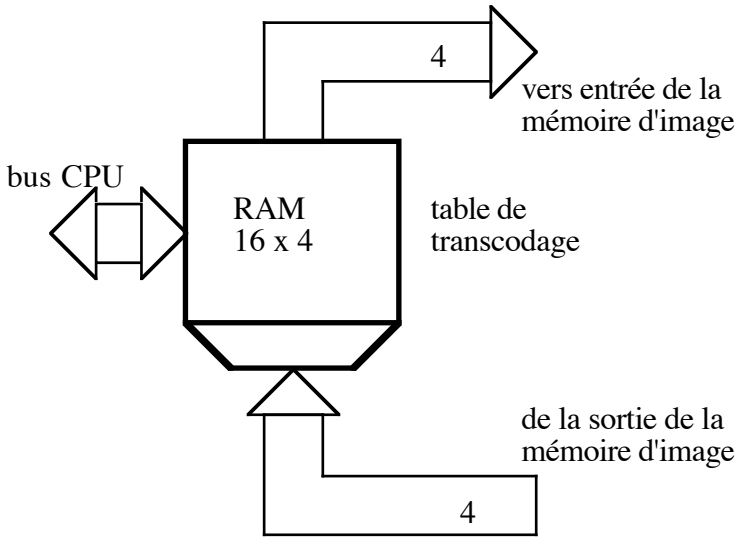

Fig. 4.5: Transcodage des pixels

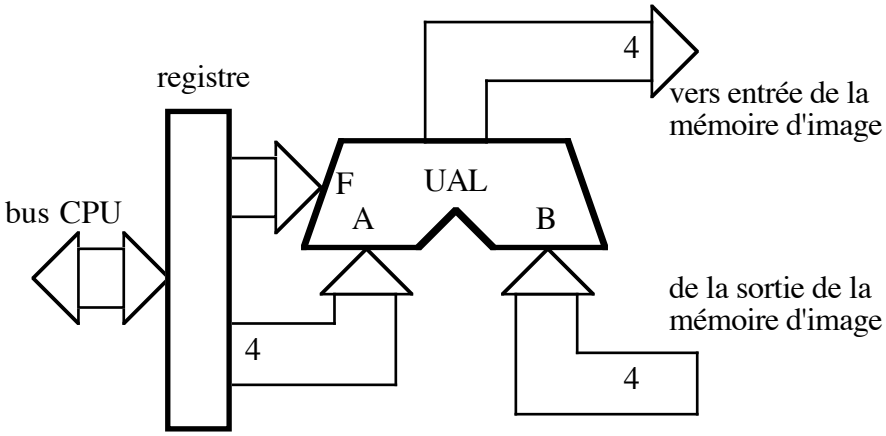

Fig. 4.6: Transcodage des pixels par une UAL

Nous avons par la suite été amenés à doubler cette table et à permettre une commutation dynamique entre les 2 en fonction d'une des sorties de la table courante (figure 4.7). Ceci est intéressant dans plusieurs applications:

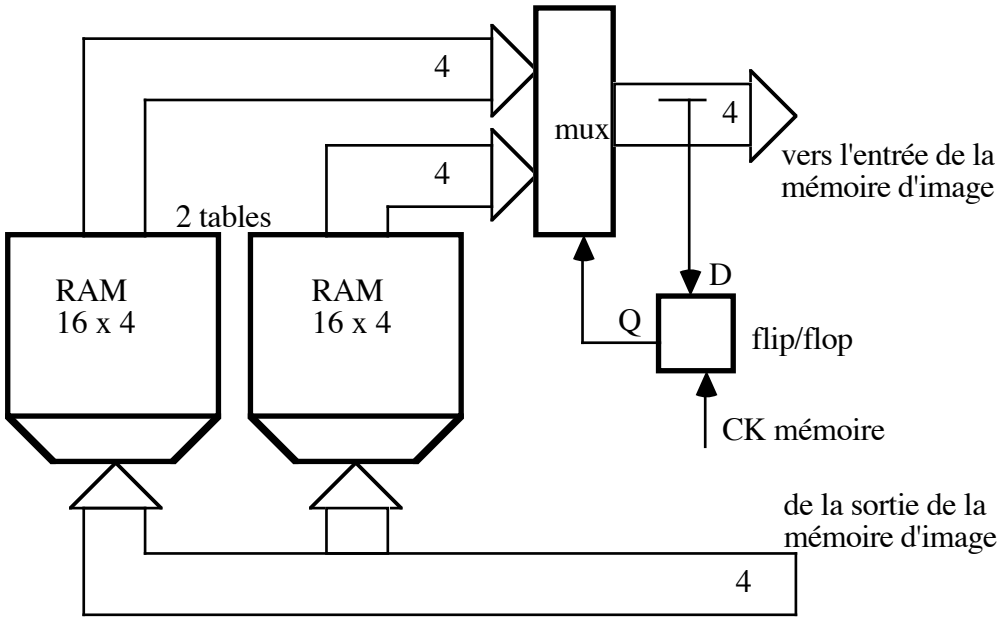

Fig. 4.7: Commutation dynamique de table de transcodage

- remplissage: (figure 4.8) on peut tracer un trait horizontal, à la vitesse du générateur de vecteurs, soit 1 M pixel / s, qui ne sera marqué qu'entre les 2 traits du contour.

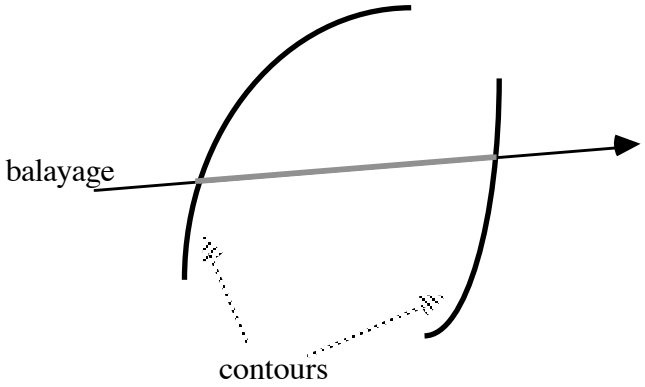

Fig. 4.8: Remplissage de zone

- détection d'objet: (figure 4.9) soit une figure complexe. On veut savoir la liste des objets proches du point (x,y). On écrit dans la mémoire d'image un code spécial pour les pixels entourant le point (x,y). On trace les objets avec une table de transcodage faisant commuter les tables si on passe sur un pixel marqué par ce code spécial. Après le tracé de chaque objet, on regarde l'état de la bascule adressant les tables.

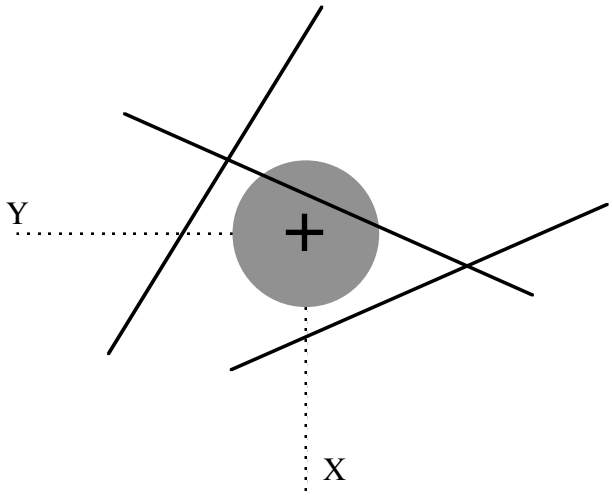

Fig. 4.9: Détection d'objet

Plan d'ensemble

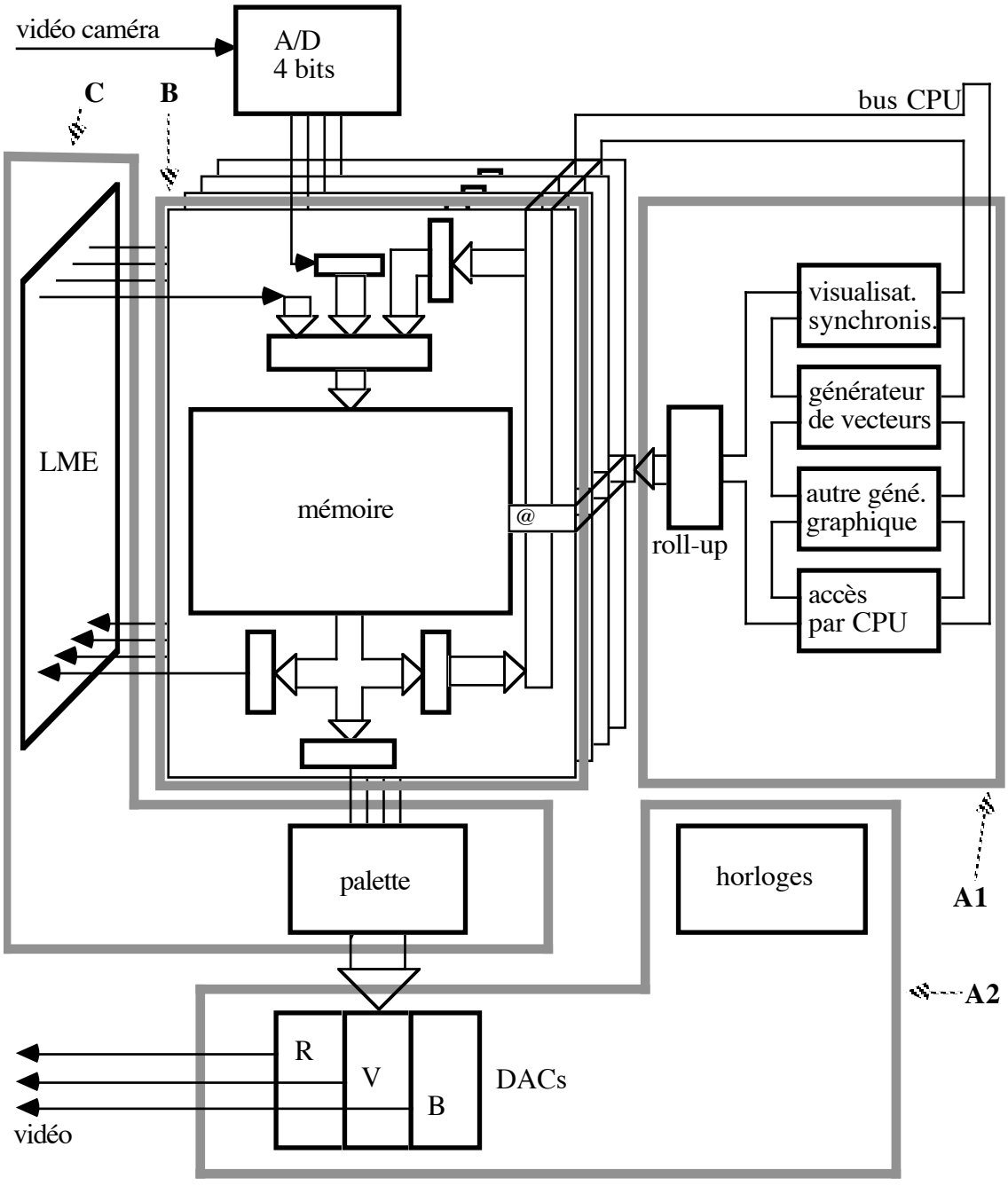

Fig. 4.10: Plan d'ensemble de la carte

On peut rassembler (figure 4.10) les différentes parties vues précédemment. Les adresses mémoire sont issues des différents générateurs (par l'intermédiaire d'un opérateur de "scroll" effectuant des additions sur les adresses). L'opérateur "accès par microprocesseur" réalise uniquement une adaptation des adresses du processeur aux adresses mémoire d'image.

Découpage fonctionnel

Du point de vue architectural, on peut découper cet ensemble de la façon suivante: (correspondant aux pointillés de la figure 4.10).

A1) Partie indépendante du nombre de bits par pixel, et à la fréquence des accès mémoire, qui regroupe tous les générateurs d'adresses:

- visualisation et synchronisation,
- tracé d'objets: vecteurs, caractères, etc ...,
- accès par microprocesseur,
- allocation de la mémoire,
- opérateur de scroll,

A2) Partie indépendante du nombre de bits par pixels, et à la fréquence de la vidéo:

- générateurs d'horloge,
- convertisseurs A/D et D/A,

B) Partie de complexité proportionnelle au nombre de bits par pixel:

- mémoire d'image,
- registres à décalage,
- registres d'accès par microprocesseur,
- circuiterie d'entrée et de sortie à 1 bit par cycle,

C) Partie très liée au nombre de bits par pixel:

- fonction de RMW (opérateur pixel),
- palette de sortie vers les DACs.

Définition d'une famille de circuits VLSI possibles

Le découpage précédent permet de dégager des fonctions assez indépendantes les unes des autres, communiquant entre elles par des bus de taille raisonnable. Ces fonctions ont une grande généralité et ceci suggère la définition d'une famille de circuits VLSI organisés ainsi:

- la partie A1 pourrait être réalisée par un circuit VLSI. Toutes ses opérations sont effectuées à la cadence des cycles mémoire. Il réaliserait l'interface entre un bus micro-processeur et l'adresse de la mémoire d'image (figure 4.11). On pourrait l'appeler "contrôleur de mémoire graphique" (CMG). Ce circuit pourrait être très complexe si les générateurs d'objets sont sophistiqués. Pour l'instant, dans notre maquette, c'est le circuit EF9365 avec quelques circuits MSI autour. Les signaux de contrôle de la figure 4.11 indiqueraient à la mémoire d'image la nature du cycle.

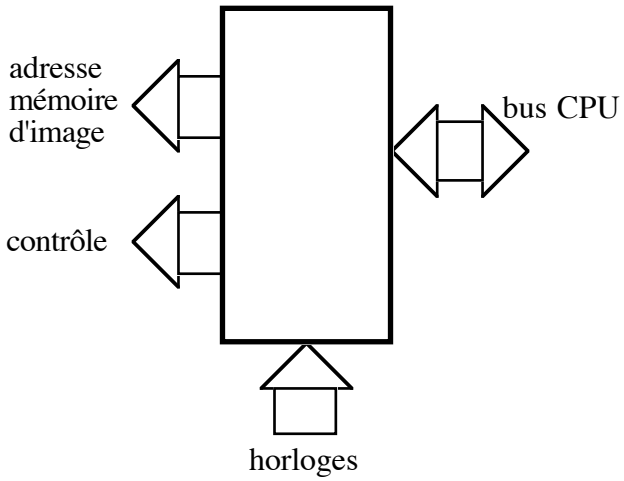

Fig. 4.11: Circuit CMG

- la partie A2 pourrait être réalisée par un LSI devant fonctionner à la cadence des pixels: circuit "horloges et conversions" (H/C) (figure 4.12).

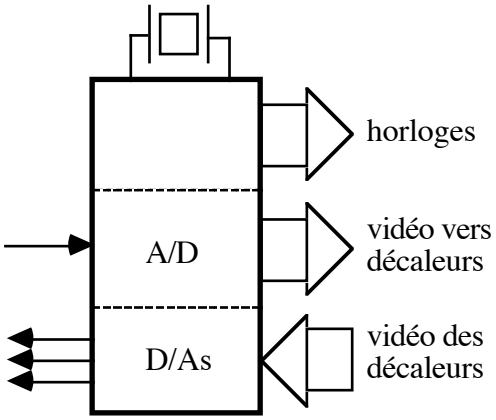

Fig. 4.12: Circuit H/C

- la partie B est une mémoire spécialisée puisque équipée d'une périphérie adaptée au graphique (MG) (figure 4.13). La capacité mémoire est 512 K bits. Une petite partie du circuit (les registres à décalage) fonctionne à la fréquence vidéo et devra être optimisée localement.

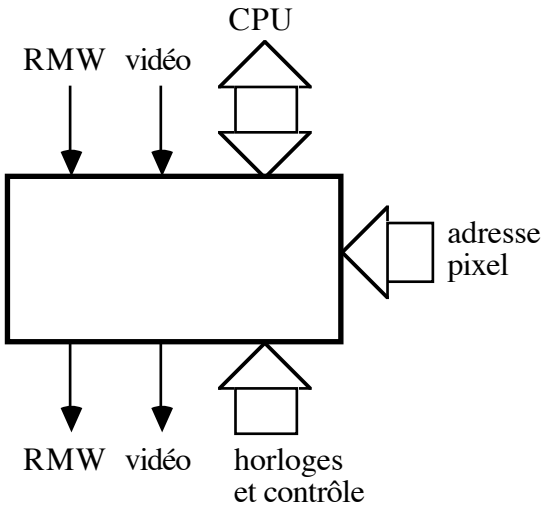

Fig. 4.13: Circuit MG

- la partie C peut être réalisée pour 4 bits par pixel, pour la CAO et le traitement de texte. Elle travaille à la fréquence vidéo (figure 4.14).

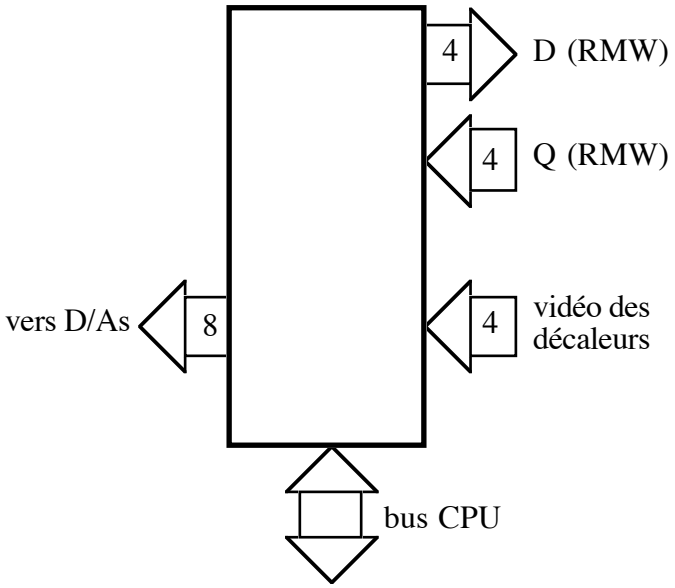

Fig. 4.14: Circuit LME/P

En définissant ainsi cette famille de circuits, on peut réaliser un contrôleur d'écran 512 x 512 et 16 couleurs en 7 circuits intégrés tenant sur une petite carte (figure 4.15).

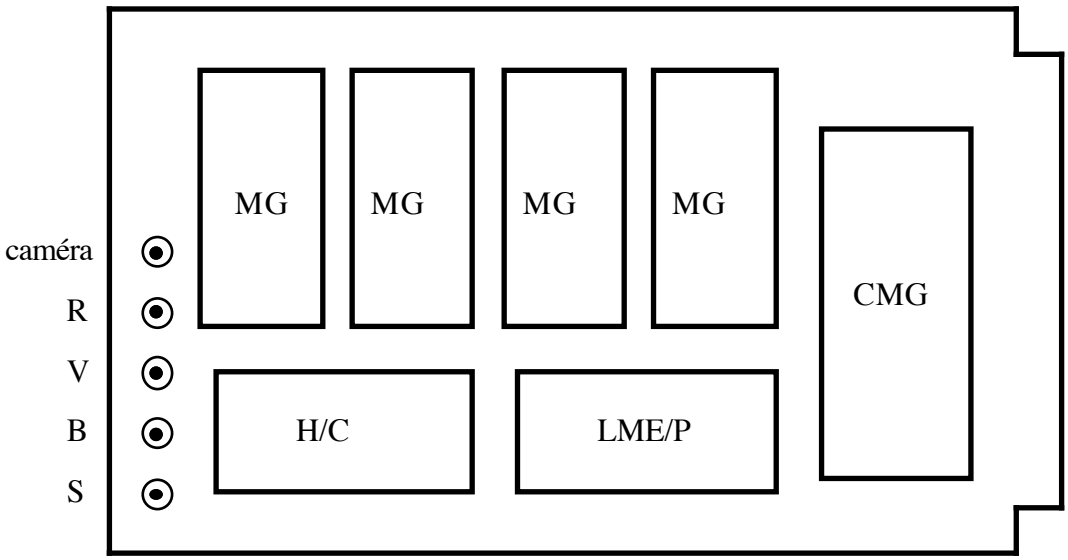

Fig. 4.15: Carte graphique VLSI

Architecture de la maquette de simulation

Nous avons implémenté cette architecture avec des composants existants. Une partie du circuit CMG est réalisée avec le circuit EF9365. Les mémoires sont des 64 K x 1 bits, il en faut donc 32. Pour garder la modularité liée au découpage précédent, nous avons découpé l'ensemble en 3 cartes distinctes:

- une carte mère rassemble les parties A1 et A2 (circuits CMG et H/C),
- une carte fille mémoire rassemble les 4 plans mémoire avec leur périphérie,
- une carte fille supplémentaire réalise le circuit LME/P.

Les cartes mémoire sont empilables de 1 à 8 grâce à un connecteur spécial. La carte LME/P doit être changée lorsque l'on modifie le nombre de cartes mémoire. La carte mère contient environ 100 circuits intégrés. La carte mémoire comporte 60 circuits. La carte LME/P contient 30 circuits (figure 4.16).

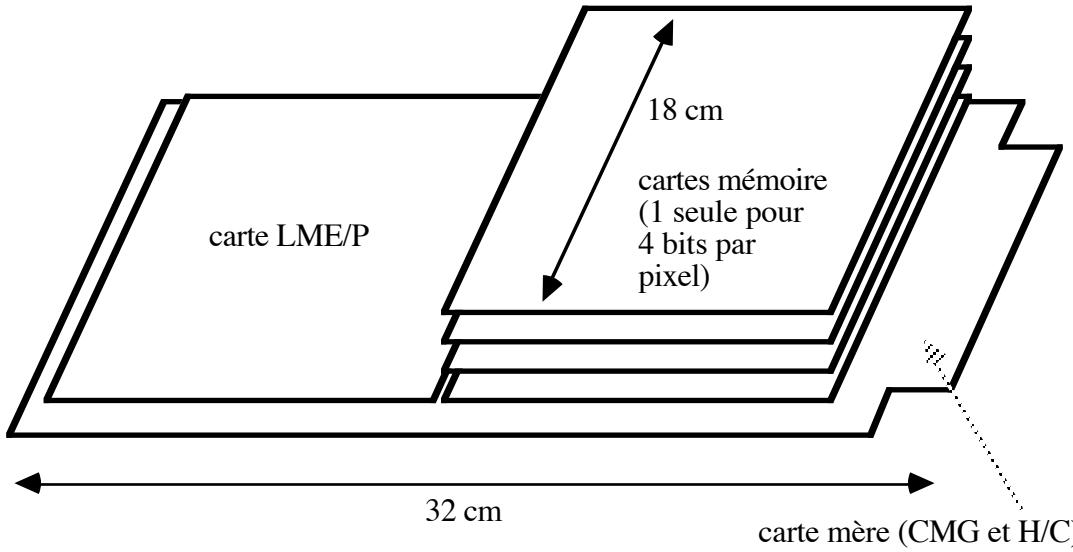

Fig. 4.16

Dans une version ultérieure plus dense (celle réalisée à 10 exemplaires), nous avons implémenté sur une seule carte (comprenant 270 composants!), à la fois la "carte mère", un exemplaire de la "carte mémoire", et la "version 4 bits de LME/P", tout en laissant la possibilité d'extension par les cartes filles précédentes.

Nous avons également réalisé la version 8 bits pour une station de traitement d'images acquises par une caméra vidéo [VPP*86].

Limitations et évolutions possibles

Le choix d'un affichage 512 x 512 n'est pas un problème. Tout ce que nous avons appris avec ce format est transposable à des résolutions plus élevées ou des formats rectangulaires. Ce choix a eu l'avantage de permettre d'utiliser le circuit EF9365, ce qui permet de remplacer 300 circuits par un seul dans la maquette. En outre, cette maquette, qui nous a appris beaucoup sur l'architecture de ces futurs circuits VLSI, réalise en elle-même une carte graphique très intégrée (de rapport performances / coût inégalé à l'époque).

Les limitations de cette architecture résident dans les 2 points suivants:

- le débit d'écriture de 1 pixel par cycle mémoire pour la génération d'objets est trop faible, en particulier pour un générateur de translation câblé, dont une application essentielle est la gestion de "fenêtres" sur l'écran,
- toutes les opérations sur un octet complet dans un bloc mémoire (visualisation par ex.) se heurtent à une frontière d'octets rigide. En particulier, un scroll horizontal câblé de précision inférieure à 8 pixels est difficile à réaliser.

Ces 2 problèmes se rejoignent. Il faut organiser l'accès mémoire plus souplement pour pouvoir accéder à des mots mémoire non forcément cadrés d'une seule manière horizontalement. Mais cela aurait beaucoup alourdi notre maquette actuelle.

Critique actuelle de ce texte

Nous reprenons donc cette fonte pour critiquer le texte précédent. Reprenons point par point:

"Choix de l'architecture globale": Le texte du premier paragraphe semble indiquer une échelle de puissance régulière, qui se gravirait en câblant progressivement davantage de primitives. Or, dans les machines qui se sont répandues, et dans les logiciels associés, n'existe qu'une primitive, d'ailleurs le plus souvent programmée, le "BitBlT". Au dessus de cette primitive, chaque gestionnaire de fenêtres, chaque système, utilise une hiérarchie de fonctions différente. Cette "échelle de puissance" n'a pas de sens, car elle suppose une hiérarchie de fonctions graphiques figée.

"Justification des choix": Nous pensons maintenant que dans le EF9365, tout ce qui est lié à l'écriture d'objets graphiques est inutile. Seul est intéressant dans ce circuit le générateur d'adresses de visualisation. Le reste n'a eu un intérêt que lorsque les microprocesseurs ne pouvaient pas adresser la mémoire d'image. En outre, dans ce circuit ou dans le reste de la carte:

- La circuiterie de photostyle est inutile, puisque la "souris" est maintenant généralisée, et ceci grâce à ses avantages certains: précision, simplicité, généralité, interactivité meilleure.
- Le "Roll-up" câblé n'a aucun intérêt dès que l'on utilise des fenêtres, car il agit sur l'image entière. On pourrait chercher à gérer les fenêtres de façon câblée, mais pourquoi ajouter de la circuiterie pour faire des choses que l'on sait faire à une vitesse suffisante par logiciel ?
- Les entrées analogiques sont inutiles dès que l'on utilise une "souris", qui est aussi préférable à un manche à balai.

"rotateur sur la donnée mémoire": ceci est fait de façon élégante par un microprocesseur tel que le Motorola 68020 qui inclut ce rotateur dans son UAL ("barrel-shifter"), utilisable pour toutes sortes d'applications par une instruction spécifique de décalage.

"situation de la carte dans l'espace d'adressage...": ce paragraphe est un début de prise en considération du Bit-map. Les difficultés citées sont justes: bande passante mémoire, taille mémoire, mais elles sont incontournables, et ces questions doivent être posées dès que

l'on aborde l'étude de l'architecture d'une visu. Outre la question de la possibilité d'adressage direct par le processeur central, on doit se poser la question de l'accès par d'autres processeurs (DMA par ex.), et à quel débit ?

"nombre de bits par pixel": aujourd'hui, on mettrait plutôt un minimum de 6 ou 8 bits par pixel.

"limitations - frontière d'octets rigide": c'est un faux problème. Deux questions seulement sont intéressantes: quel est le débit de la mémoire ? et à quelle vitesse le processeur fait-il les décalages ? Le débit de la mémoire doit être suffisant pour la visualisation et la lecture-modification-écriture du BitBlT. Les décalages sont faits rapidement par un microprocesseur moderne.

Points positifs

Signalons après toutes ces critiques, que cette carte comporte des originalités, ou des particularités qu'on rencontre rarement, et qui lui gardent un intérêt, encore aujourd'hui:

- le couplage étroit avec l'acquisition d'une image caméra,
- le paramétrage de la palette comportant moitié niveaux de gris et moitié couleurs, permettant d'annoter en couleur une image N&B, ou tout simplement de ne pas changer le moniteur entre 2 programmes d'application,
- le balayage télévision permettant, en liaison avec un magnétoscope, de synthétiser une bande magnétoscope image par image,
- et surtout l'opérateur pixel pour le transcodage en vol: remplissage de contour et détection d'objets.

Conclusion

A la suite de ce changement de point de vue, nous avons fait l'expérience de refaire le schéma de cette carte, en s'orientant vers une utilisation plus large du BitBlT sur un microprocesseur puissant, et en enlevant tout ce qui est cité comme "inutile" dans le paragraphe de critique: roll-up câblé, photostyle, manche à balai, synchronisation pour une visu alphanumérique, tracé de pointillés, opérateur pixel, opérateur arithmétique virgule flottante.

Nous avons alors constaté que la complexité du schéma est divisée par plus de deux, l'essentiel de la circuiterie restante étant due à la modularité du nombre de bits par pixel.

La conclusion de cette étude est que la caractéristique très spéciale d'une visu graphique, par rapport à un autre périphérique, est l'association entre:

- un débit très élevé,
- et une horloge précise imposée par le balayage vidéo.

Pour résoudre ce problème, il faut désynchroniser l'horloge de la mémoire de l'horloge de la vidéo, ce qui n'est possible que si la mémoire permet un débit bien supérieur à ce qui est demandé par la vidéo (au moins 2 à 3 fois), et à partir de ce moment, la visualisation devient un DMA classique.

Ceci nous a conduit à un projet basé sur une architecture très différente, décrite dans les chapitres 6 et 7.

Chapitre 5: Architectures spécialisées pour la synthèse d'image

Historiquement, les premières machines implémentant de façon câblée des algorithmes de synthèse d'image ont été réalisées pour les "simulateurs de vol" permettant d'entraîner les pilotes d'avions au sol. Ces machines ont été réalisées très tôt (dès 1962), probablement parce que le coût de développement élevé n'est à comparer qu'au coût de fonctionnement des avions de chasse, voire même à celui du lancement d'un satellite artificiel pour les machines de simulation utilisées à la NASA.

La qualité des images n'était alors pas très grande par rapport à ce que nous connaissons maintenant sur des appareils relativement bon marché. Dans le système développé par GE en 1965 pour la NASA, on affichait sur un écran télévision des facettes en niveaux de gris sur 10 bits, à l'aide d'une machine comprenant 140 000 circuits [Sch81]. Les opérations indispensables sont: calcul de perspective, clipping, élimination de parties cachées.

Le temps réel était bien entendu la priorité numéro un. Nous avons signalé dans le premier chapitre les pipe-lines utilisés et les structures de processeur d'affichage associées. La deuxième priorité était l'efficacité de l'apprentissage, c'est-à-dire la rapidité de reconnaissance par le pilote, ce qui n'est pas toujours la même chose que le réalisme, dans la mesure où ces scènes étant toujours du même type, certains objets peuvent être représentés de façon symbolique. Au fur et à mesure que la technologie le permettait, on a ajouté à ces machines les qualités de rendu ou la complexité d'image compatible avec le temps réel.

Ce n'est que très récemment (années 1980) que ces techniques ont été plus largement utilisées, dans les "stations de travail graphiques".

La station IRIS

La première réalisation (1979-1982) a été celle de James Clark qui a adapté la chaîne de pipe-line classique [CH80a et b] [Cla80a et b] :

- en réalisant des circuits VLSI spécialisés,
- en calculant des polygones en couleurs,
- et en effectuant les calculs en virgule flottante.

Cette étude a débouché sur un produit commercial: l'IRIS de la société "Silicon Graphics". Dans cette station, la partie spécialisée effectuant la synthèse d'image est constituée de 12 circuits VLSI réalisant le pipe-line, 4 en parallèle effectuent les multiplications de matrices 4 x 4 puis 6 en série se chargent du clipping par les 6 plans de la pyramide de vision, puis 2 en parallèle font la mise à l'échelle finale. Ces 12 circuits sont en fait des configurations légèrement différentes d'un seul, appelé par Clark "Geometry Engine". Chacun de ces circuits contient 4 unités flottantes identiques sur 28 bits (comprenant chacune un UAL, 3 registres et une pile), capable de réaliser des additions, soustractions et décalages). Dans la version de 1982 [Cla82], la première commercialisée, ce pipe-line effectue les transformations géométriques à un débit de 80 k points par seconde, ce qui correspond à une puissance de calcul de 4 MFlops (48 unités à 12 μs). A la sortie du pipe-line se trouve une mémoire d'image organisée en Z-buffer pour résoudre les parties cachées. La vitesse a été augmentée dans les versions plus récentes.

CUBI7

A la même époque, la machine CUBI7 [Ler83], étudiée par l'équipe de P. Leray au CELAR, implémente avec des circuits standards (3 cartes de 300 circuits), un pipe-line voisin, réalisant en outre un lissage sur les facettes (méthode de Gouraud). La synthèse s'effectue en deux temps:

- une partie de préparation sur calculateur classique calcule des "priorités statiques" entre les objets de la scène suivant la méthode de Schumaker. Cette partie hors temps réel peut prendre quelques dizaines de minutes.

- les transformations géométriques, les calculs d'éclairement (ombrage) et la préparation au lissage sont effectués en temps réel par un processeur à base d'Amd 2900 qui cycle à 200 ns. Le traçage effectif avec lissage est ensuite effectué à la vitesse de 100 ns par pixel dans 2 mémoires d'image en "double buffer".

Les performances indiquées en 1980 [Ler80] sont: 200 facettes par processeur en temps réel pour des images 512 x 512 x 8 à 25 Hz.

GETRIS

Une autre machine, commercialisée en 1985 par la société GETRIS-Image a été mise au point par l'équipe de Martinez au Laboratoire Artémis [Fer81] [MF81]. Cette machine ne traite pas les transformations géométriques, mais s'occupe particulièrement du rendu final:

- position des sources lumineuses,
- textures,
- réflexion diffuse et spéculaire.

L'idée de base est d'effectuer tous les calculs en aval de la mémoire d'image, pixel par pixel, à la fréquence vidéo. La mémoire d'image doit contenir:

- la profondeur Z,
- le numéro de chaque facette,
- l'orientation de la normale, etc,

et ceci pour chaque pixel.

Cette méthode permet par exemple, sur une scène fixe, de déplacer les sources lumineuses en temps réel, quel que soit la complexité de la scène. En 1985, la machine calcule un lissage de Phong en temps réel.

Machines pour le lancer de rayon

Nous venons de citer trois réalisations basées sur des pipe-lines à travers lesquels transitent des polygones (ce n'est pas tout à fait le cas de la machine GETRIS, mais elle est adaptée au post-traitement d'images décrites par des polygones). La synthèse d'image par "lancer de rayon" demande elle un parallélisme très différent. Il est plus naturel d'effectuer dans des processeurs complètement disjoints les calculs associés à des pixels différents. On aboutit à une structure multiprocesseurs travaillant en MIMD.

Citons par exemple la machine CRISTAL du CCETT [Bru86]. Chaque processeur (appelé CPIX) comporte, sur une carte, un processeur NS32000 à 10 MHz avec son co-processeur flottant et une mémoire pouvant aller jusqu'au M octet. On peut installer jusqu'à 128 processeurs (on a alors 8 châssis de 20 cartes). Avec 40 processeurs, on peut calculer une image de complexité moyenne en 10 minutes, ce qui correspond à une puissance de 1 GFlops.

Ces dernières années, les progrès sur ce type de machine ont été impressionnants. En Juillet 1986, la société MEICO a présenté une machine à base de 300 Transputers. Chacun de ces processeurs, développés par INMOS, a une puissance de 10 MIPS. Une image représentant 3 sphères, un échiquier et 3 sources lumineuses est calculée en moins de 2 minutes. En juillet 1987, AT&T a présenté une machine sur le même principe, mais utilisant des processeurs développés pour l'occasion, calculant une telle image environ 10 fois plus vite.

Machines cellulaires

Une troisième approche est celle de H. Fuchs qui réalise une mémoire d'image intelligente en plaçant pour chaque pixel un processeur très simple et très spécialisé. Cette optique permet un plus grand nombre de processeurs, dans une structure SIMD [FP80] [FP81] [FPPB82] [FGH*85] [PFA*85].

La première version, appelée "Pixel-planes", traite les faces cachées, ainsi que les ombrages et lissages (de Gouraud) en calculant une approximation linéaire avec 2 additionneurs par pixel. La mémoire - matrice de processeurs est alimentée avec les intensités R, V, B, Z aux sommets des polygones, et chaque pixel calcule son intensité. Le temps de calcul est indépendant de la dimension du polygone. Les performances en 1981 étaient (avec des circuits fonctionnant à 5 MHz et comprenant 4 pixels chacun!) de 500 quadrilatères en temps réel.

Dans la deuxième version, appelée "Pixel-power" [GF86] [GHF86], les processeurs élémentaires peuvent calculer des fonctions quadratiques. L'application est le calcul d'intersection de volumes tels que cylindre, cônes, etc pour résoudre les parties cachées et l'ombrage d'images décrites par un arbre CSG. Les performances sont: 24 primitives CSG (1900 équations quadratiques) calculées en 7,5 ms. Dans le même esprit, on peut citer [SII*85].

Réflexion sur ces machines

Les machines que nous venons de décrire cherchent toutes à réaliser un meilleur compromis entre le temps de calcul des images et la qualité finale de celles-ci (réalisme). Les progrès dans les algorithmes de synthèse montrent la possibilité de calculer des images de plus en plus surprenantes:

- soit par la complexité ou le nombre des objets représentés: scènes de plusieurs dizaines de milliers d'objets, objets procéduraux tels que montagnes ou arbres fractals, systèmes de particules tels que fumées, feux d'artifices, etc

- soit par la complexité des phénomènes lumineux pris en compte: réflexion, réfraction, diffusion, réflexion spéculaire, éclairage indirect, etc

- soit même par la complexité des modèles physiques décrivant les relations entre les objets entraînant leurs mouvements.

Chaque profession d'utilisateurs est intéressée par un aspect particulier, et utilisera une méthode de synthèse particulière. Les images les plus belles, les plus réalistes ou utilisant les effets spéciaux les plus innovateurs sont utilisées dans le domaine de la publicité ou du cinéma, où les coûts sont à comparer à ceux de la prise de vue, soit quelques dizaines de kF par seconde. On a intérêt alors à utiliser l'architecture d'ordinateurs la plus souple possible, avec toutes les possibilités d'un environnement de programmation performant. On peut utiliser les ordinateurs généraux les plus puissants, qui sont aussi les plus gros (dimensions physiques, dissipation thermique, prix). Le temps réel n'est pas une contrainte, mais on cherche à diminuer le temps de calcul pour diminuer la difficulté de mise au point des séquences. Il est en effet intéressant de réduire de 10 heures à 15 minutes le temps de calcul d'une image, en utilisant un supercalculateur!

Les stations de travail utilisent des circuits spécialisés pour rendre plus abordable cette puissance de calcul. Il faut abaisser le coût à moins de 1 MF, pour des images de qualité maximale compatibles avec l'interactivité (temps de calcul de l'ordre de la seconde ou moins).

Il est clair que le seul problème (mais il est de taille!) est la puissance de calcul réalisable sur une machine de dimension et de coût réduits. Il n'y a pas d'autre nécessité qui pousse à ces architectures spécialisées. Et cette spécialisation apporte de fortes contraintes. Une machine développée pour un type d'algorithme sera peu efficace pour des algorithmes très différents, et pourtant le temps et le coût de développement sont très importants. Il est très risqué d'investir des dizaines ou des centaines d'hommes x années dans une machine dont la durée de vie sera de 2 à 3 ans.

Ceci nous pousse à chercher une solution dans la direction des architectures générales. Toutes les machines citées ont en commun un fort parallélisme, mais spécialisé. Comme par ailleurs les progrès dans l'utilisation du parallélisme sont prometteurs, il nous semble intéressant de regarder si le facteur de puissance recherché ne pourrait pas être obtenu par une machine fortement parallèle, mais tout de même de taille et de coût raisonnables.

Chapitre 6: Une visu sans mémoire d'image spécialisée, sur un ordinateur à bus modulaire

Présentation

Nous décrivons dans ce chapitre une réalisation récente, mise en oeuvre sous notre direction par Bruno Liège, à l'occasion de son DEA [Lie86]. Il s'agit d'une carte de visualisation 800 x 1000 à 60 Hz monochrome, sans mémoire d'image spécialisée, installée sur un ordinateur à base de microprocesseur 32 bits du commerce et organisée autour d'un bus modulaire. Le processus de visualisation est un DMA classique sur la mémoire générale.

Cette architecture nous a été suggérée par les remarques faites en analysant les machines citées dans les chapitres précédents:

- 1) Les architectures avec mémoire d'image spécialisée ont été utilisées pour résoudre la contrainte de débit insuffisant du bus mémoire de l'ordinateur hôte dans les structures à base de microprocesseur standard,
- 2) Les architectures avec co-processeur graphique ont permis de créer un parallélisme de traitement dès lors que la mémoire d'image était séparée de la mémoire générale. Ce parallélisme à 2 processeurs de puissances comparables est mal utilisé et n'est pas très satisfaisant puisque la puissance des microprocesseurs généraux augmente très rapidement avec l'évolution technologique,
- 3) Les logiciels graphiques utilisent intensivement la primitive BitBlT dont l'efficacité est meilleure sur les processeurs généraux en mémoire générale.

Signalons que le montage que nous décrivons dans ce chapitre est très proche de celui de la visu de l'Alto, la différence résidant dans le fait que le DMA est ici câblé et indépendant du processeur central, alors que dans l'Alto, il est microprogrammé. C'est en quelque sorte une réconciliation de 2 voies architecturales opposées:

- les machines de Xerox-PARC (Alto, Dorado, ...) sur lesquelles le BitBlT a été inventé, et qui ont exécuté les gestionnaires de fenêtres avec la meilleure efficacité, n'utilisaient pas de microprocesseur mono-circuit du commerce, ni de bus modulaire multi-maître, et simplifiaient la circuiterie en partageant le processeur en micro-tâches, induisant le partage de la mémoire,
- les machines qui se sont répandues ont au contraire utilisé des microprocesseurs standard et des bus modulaires pour diminuer les coûts de mise au point et d'adaptation (bus Europe 8-16 bits, puis Multibus, VME-bus, etc).

Il y avait donc une opposition totale entre ces 2 architectures. Une première conciliation a pu être réalisée par le Macintosh, au prix de l'abandon de la modularité du bus : certes on partage la mémoire entre des périphériques divers, mais d'une façon décidée à la conception. Tous les cycles sont pré-alloués et la machine est fermée.

Le seul paramètre dont la modification peut réunir ces architectures est le débit du bus modulaire. Un écran 800 x 1000 x 60 Hz consomme en moyenne 6 M octets/s. S'il s'agit d'un écran couleur 8 bits, on passe à 48 M octets/s. Il faut donc que les bus passent de quelques M octets/s à quelques dizaines de M octets/s.

Il faut bien voir que tout ceci est indépendant de la complexité des images synthétisées. En effet, dans le cas d'images entièrement animées, on calcule une image entièrement nouvelle pour chaque image affichée sur l'écran. Le débit demandé à la mémoire pour l'affichage est dans ce cas au pire égal à celui demandé par le processeur qui synthétise l'image, et généralement beaucoup plus faible.

Si la synthèse demande un débit beaucoup plus élevé que la visualisation, alors ce n'est pas cette dernière qui est critique, et si inversement la synthèse est très simple, négligeable devant le débit de visualisation, il suffit que le débit du bus soit supérieur à ce dernier.

En résumé, il n'y a aucune raison de mélanger le problème de la visualisation et celui de la synthèse. Une fois résolu le premier, le deuxième est un problème général de puissance du processeur, et les structures à mettre en place pour le résoudre sont d'un autre ordre que celui de quelques astuces d'aménagement de la visualisation.

L'argument selon lequel un co-processeur graphique est indispensable pour accélérer la génération des images est complètement déplacé. Si l'on a besoin d'un processeur 10 à 100 fois plus puissant pour calculer des images complexes et permettre ainsi l'interactivité, ce n'est pas en rajoutant un co-processeur de la même puissance que le processeur central, avec tous les problèmes d'échange et de synchronisation que cela suppose, qu'on résoudra le problème. Si on rajoute un co-processeur graphique 10 à 100 fois plus puissant, alors pourquoi ne pas l'appeler processeur central et supprimer l'autre?

Cette structure avec un co-processeur est issue d'une époque où l'on était obligé d'en ajouter un pour gérer une mémoire spécialisée, introduite elle-même par ce problème de débit du bus.

Notre propos n'est pas de nier le problème de puissance du processeur de synthèse d'image. Il s'agit bien du problème central, et nous y chercherons une solution dans le chapitre suivant.

Description du cadre de la réalisation

Nous avons démarré ce projet à une époque où l'équipe du Laboratoire d'Informatique de l'ENS était engagée dans un projet de conception d'un ordinateur individuel intégré du type "station de travail". L'originalité de l'approche de notre équipe résidait dans le choix de maîtriser un bus modulaire rapide de partage de la mémoire entre la (ou les) unités centrale (s) et les périphériques.

Nous avions réalisé un premier prototype d'expérimentation appelé M0. Le bus est synchrone à la mémoire, et un allocateur (à priorités figées) attribue chacun des cycles aux processeurs, aux DMAs, etc. en fonction de l'arrivée des requêtes [Cue84].

Dans cette architecture, la mémoire est le "centre" de la machine, et son partage est le mode de communication entre les différents organes. Le bus est donc l'organe qui assure ce partage, "en multiplexant" la mémoire. Les cycles du bus utilisés pour les échanges directs entre 2 processeurs, ou entre un processeur et un périphérique, sont en proportion négligeable.

Ce bus est synchrone, 32 bits, multiplexé adresses / données. A chaque cycle, sur chaque fil du bus, est d'abord positionnée une adresse par le maître de l'échange, puis dans la deuxième partie du cycle, la donnée correspondante est échangée (figure 6.1).

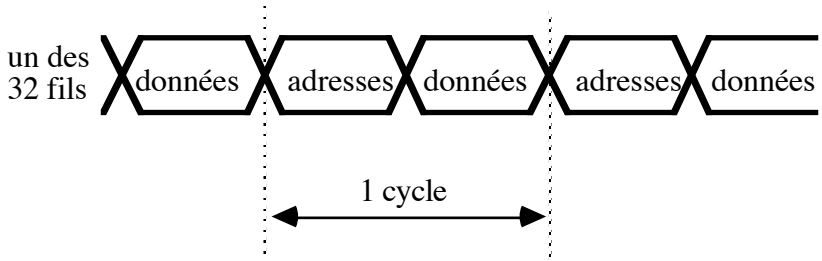

Fig. 6.1: Bus synchrone multiplexé

La durée du cycle, dépendant en partie du temps d'accès mémoire, et en partie de la réalisation du fond de panier, a été fixée à l'époque à 280 ns (3,5 MHz). Ce temps assez long est dû principalement à la réalisation en "wrapping" du fond de panier dans ce prototype, car les mémoires avaient un temps de cycle de 100 ns. Ceci permet tout de même un débit de 14 M oct/s.

La carte de visualisation 800 x 1000 à 60 Hz demande un débit moyen de lecture de la mémoire de 6 M octs/s. Elle laisse donc 8 M octets/s pour les autres maîtres (processeur, disque, Ethernet), ce qui est satisfaisant, surtout si l'on considère le caractère d'expérimentation de cette réalisation.

Description de la carte de visualisation

Cette carte est donc un DMA de lecture mémoire vers le moniteur vidéo. Comme l'horloge vidéo (liée au moniteur) n'a aucune raison d'être synchrone avec l'horloge de la mémoire et du bus, il est nécessaire d'intercaler un FIFO sur le chemin de données entre le bus et la vidéo (figure 6.2). La longueur de ce FIFO doit être choisie en considérant 2 choses :

- 1) La vidéo ne porte pas d'information dans les marges de l'image. Par contre, lors de l'affichage d'une ligne, le débit est environ le double du débit moyen (10 ns par pixel = 12 M octets/s). Si la carte visu pouvait répartir régulièrement ses demandes sur le bus, il faudrait un FIFO contenant 1/2 ligne, soit 1000 pixels / 2 / 8 = 64 octets. Ainsi, si le FIFO

est vide en fin de ligne, il se remplie pendant le retour du spot vers le début de la ligne suivante, et se vide linéairement pendant la visualisation de la nouvelle ligne.

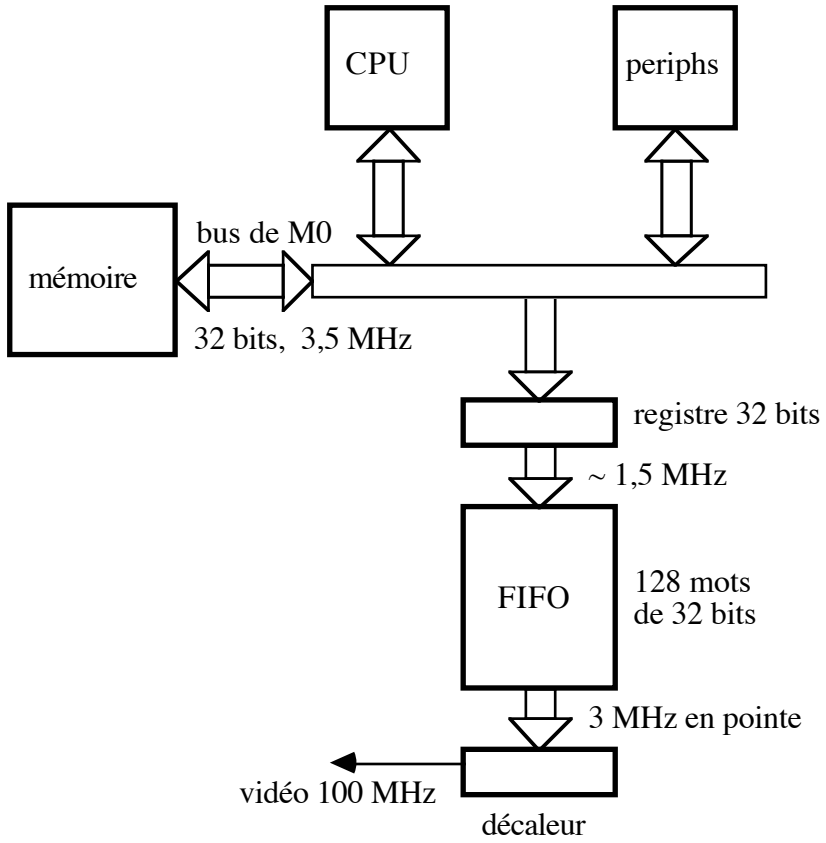

Fig. 6.2: Schéma de principe de la visu

- 2) En fait, cette carte ne peut pas avoir la plus grande priorité sur le bus si l'on ne veut pas perdre des messages Ethernet ou des secteurs disques. Il faut donc prévoir un FIFO plus grand. La taille exacte de celui-ci ne peut être choisie qu'en considérant la structure des autres contrôleurs existants (et à venir si l'on veut convaincre de la modularité du bus), principalement la longueur des FIFOs de ces contrôleurs. Disons simplement que l'on trouve facilement des circuits intégrés réalisant des piles FIFO de profondeur 512 ou 1 K, et que c'est satisfaisant pour notre application.

Ceci nous conduit au schéma de la figure 6.2 pour le chemin de données.

A celui-ci, il faut ajouter :

- 1) L'horloge vidéo et un diviseur par 32, pour décaler et charger le registre à décalage,
- 2) Un séquenceur pour générer le signal de synchronisation du moniteur,

- 3) Un générateur d'adresses pour aller chercher les données en mémoire,

- 4) Un contrôleur pour déclencher les demandes de cycles sur le bus en fonction de l'état de remplissage du FIFO.

Le point 1 est simple à régler : (figure 6.3) L'horloge à 100 MHz et le diviseur par 32 doivent tourner constamment pour séquencer de façon synchrone toutes les opérations concernant le moniteur vidéo.

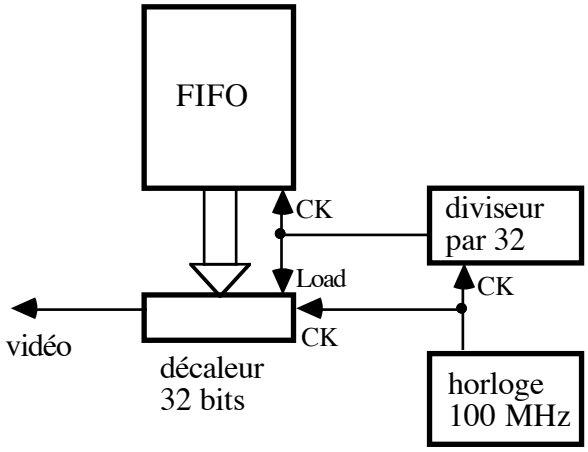

Fig. 6.3: Séquencement de la sortie du FIFO

On remarque immédiatement que ce montage oblige à faire passer des données nulles dans le FIFO lors des marges.

Une autre solution consisterait à inhiber le chargement du décaleur (et la sortie du FIFO) lors de ces marges.

Le choix entre ces 2 solutions est lié au choix de l'horloge du séquenceur générant la synchronisation du moniteur (point 2). En effet, soit ce séquenceur est activé par l'horloge vidéo (synchrone donc avec la sortie du FIFO) et il inhibe ainsi le décalage lors des marges; soit ce séquenceur est synchrone avec l'horloge du bus de la machine (et de l'entrée du FIFO) et le décaleur fonctionne constamment, obligeant à entrer des données nulles dans le FIFO lors des marges. Dans ce cas, le signal de synchronisation du moniteur doit lui-même traverser le FIFO, pour se trouver synchrone à la vidéo.

Bruno Liège a choisi cette deuxième solution, qui permet plus facilement les changements de moniteur, ceci concernant autant le générateur d'adresses (point 3) que le générateur de synchronisation.

Avant de passer à ces générateurs, notons que l'on peut simplifier ce schéma en réduisant à 8 bits la largeur du FIFO, et en multipliant par 4 sa longueur. Il faut ajouter un bit à la largeur pour le signal de synchronisation. Le schéma devient celui de la figure 6.4.

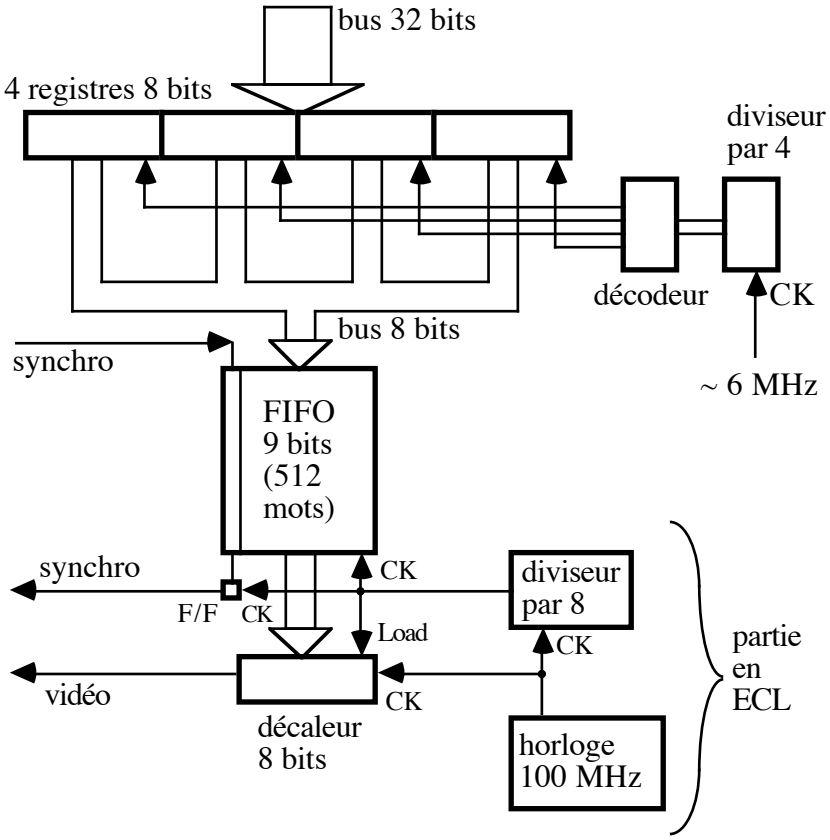

Fig. 6.4: Chemin de données complet

Séquenceur vidéo et générateur d'adresses

Il s'agit essentiellement de 2 compteurs, dont l'horloge est synchrone avec le bus, mis à part le fait que celle-ci s'arrête quand le FIFO est plein.

Le séquenceur vidéo doit être paramétré par le format d'affichage sur le moniteur (dimension du tableau de pixels affiché, fréquence ligne, fréquence trame, position du coin supérieur gauche de l'image sur l'écran).

Le générateur d'adresses lit séquentiellement en mémoire tous les mots de 32 bits nécessaires pour l'image, à partir d'une base fixée par un registre initialisable par le processeur.

On peut donc réaliser ces 2 fonctions grâce à un compteur programmable, un additionneur, et un registre (figure 6.5). On remarquera que la simplicité de ce schéma est liée au choix de rendre systématique le chargement du registre à décalage.

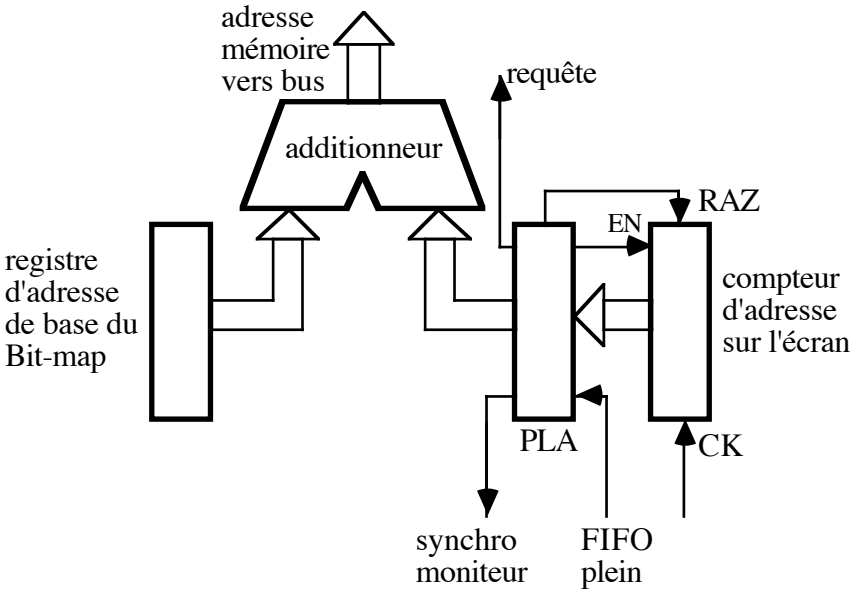

Fig. 6.5: Séquenceur et générateur d'adresses

Le PLA est le seul élément à modifier lorsque l'on change le format d'affichage.

Réalisation physique de la carte et conclusions

B. Liège a réalisé cette carte (dans un schéma très légèrement différent). Le montage comporte un peu moins de 50 circuits intégrés classiques, dont près de la moitié sont indispensables pour l'adaptation au bus!

Peu de visus de cette performance sont réalisées de façon si simple. On remarquera également la minimisation de la partie en ECL tournant à 100 MHz.

Malheureusement, l'étape indispensable de vérification de l'efficacité d'un logiciel complexe tel qu'un gestionnaire de fenêtres n'a pu être réalisée car Unix n'a jamais été installé sur la machine M0, en raison de l'arrêt du projet global, lié au départ d'une partie de l'équipe.

Nous aimerions reprendre cette expérimentation sur une autre machine, car outre la simplicité de l'architecture, nous voyons un intérêt particulier à celle-ci:

Dans tous les montages avec mémoire d'image spécialisée, cette dernière est synchrone avec l'horloge de la vidéo, et son séquencement est très lié au format d'affichage. Il est donc très difficile de changer notablement celui-ci, et impossible d'afficher simultanément une même image sur 2 écrans de balayages très différents.

Par contre, avec notre montage, il est facile d'afficher dans une fenêtre d'un écran 1000 x 1000 à 60 Hz, l'image d'une caméra vidéo 500 x 800 à 40 Hz par exemple, et en temps réel. Il suffit de disposer d'un DMA semblable en entrée vidéo sur la même mémoire générale. On peut ainsi effectuer des conversions de standards en rajoutant uniquement une carte de 50 circuits pour chaque entrée ou sortie vidéo. Il faut bien entendu que le débit total du bus (et de la mémoire) le permette, mais cela devient presque classique avec plusieurs bus qui apparaissent sur le marché autour de 100 M octets/s.

Chapitre 7: Une petite machine massivement parallèle

Présentation

Rappelons notre démarche: nous souhaitons réaliser des machines de visualisation d'images performantes. Nous avons montré, dans les chapitres précédents, que la partie visualisation proprement dite ne doit pas être rendue dépendante de la partie synthèse. La première est un flot constant entre la mémoire de la machine et l'écran vidéo. La seconde demande une grande puissance de calcul pour exécuter les programmes de synthèse.

Dans ce domaine des algorithmes de synthèse d'image, les efforts accumulés depuis plus de 20 ans par de nombreuses équipes montrent qu'il n'est pas possible de réduire toutes ces méthodes en un seul algorithme universel qu'il serait souhaitable de figer dans une circuiterie spécialisée. Autant d'objets à visualiser, autant de modèles et autant d'algorithmes. Certes les calculs de perspective, de clipping et de mise à l'échelle sont communs à toute visualisation 3D, mais ils représentent finalement une partie relativement faible des calculs dans les méthodes telles que le "lancer de rayon" ou la "radiosité".

Il nous semble que toute machine spécialisée implémentant de façon très rapide un algorithme particulier aura un succès commercial, car elle facilitera la vie d'une catégorie professionnelle à qui elle sera adaptée, mais sera, à terme d'un ou deux ans, supplantée par des machines plus puissantes, alors que son développement aura demandé des efforts considérables. Nous pensons qu'actuellement, l'accroissement annuel de puissance des stations de travail est tel qu'il enlève tout intérêt à la conception de machines spécialisées.

Concevoir une machine générale très puissante

Notre problème de synthèse d'image rejoint donc un problème très général, puisque dans tous les domaines, on demande toujours davantage de puissance de calcul. La particularité du graphique réside dans le besoin que l'organe de calcul soit petit et individuel, en liaison étroite avec la station de travail. Si l'on veut calculer 25 images / seconde pour une animation, il est impensable de partager le temps d'un serveur de calculs situé dans un autre bâtiment, et utilisé par 10 autres personnes. Cet organe de calcul doit donc être petit et relativement bon marché.

Ces considérations semblent d'une grande exigence. Tout le monde rêve de machines extrêmement puissantes, petites et bon marché. Pourtant, jusqu'à présent, les efforts pour la conception de machines très puissantes vont vers les "grosses" machines (cherchant la puissance, indépendamment des autres facteurs) [Electronics du 23-6-86]:

- CRAY X-MP: 400 MFlops, 10 M Dollars [Rus78],
- Connection-Machine CM2: 8 000 MIPS, 3500 MFlops, 28 kW, 3 M Dollars [Hil85],

Cette tendance se modifie, avec les "Crayettes", ou les machines parallèles telles que:

- Intel iPSC-VX, jusqu'à 1000 MFlops, 850 k Dollars [Sei85] [Electronics du 14-4-86],
- FPS, machine très comparable [Fre86],

mais il s'agit toujours d'armoires dans une salle climatisée.

Et pourtant, le paradoxe est grand lorsqu'on s'aperçoit que le principal facteur responsable de la difficulté d'augmenter la puissance des machines est la dimension géométrique de celles-ci. En effet, des différentes limites imposées par la nature à la puissance des machines, la *seule* atteinte actuellement est celle de la vitesse de la lumière. Cette dernière "se traîne" à 30 cm/ns. Or, nos cartes de circuits imprimés font 30 cm, et nos portes logiques (dans les CMOS - VLSI) commutent toutes les ns.

Par contre, nous savons miniaturiser: nous pouvons intégrer plusieurs millions de transistors par cm^2 de Silicium. Nous savons aussi accélérer la commutation: les portes As-Ga ou Josephson commutent en quelques 10 ps, et la supraconductivité à température ambiante ou presque n'est plus pour demain.

Or, dans une machine actuelle:

- s'il s'agit d'une machine série (Von Neumann), quand nous allons chercher un opérande en mémoire, si celle-ci se trouve à quelques centimètres sur la même carte (ce qui est rare, en général, il faut passer 2 connecteurs pour aller sur une autre carte), alors nous acquittons le péage à la lumière: quelques nanosecondes.

- s'il s'agit d'une machine très parallèle telle que la "Connection-Machine" (CM1 de TMC): son but est de multiplier les possibilités de communication. Ses messages peuvent transiter par 2^{12} arêtes de l'hypercube. Mais l'équivalence logique entre ces arêtes cache une différence physique: certaines ont une longueur de quelques cm, d'autres quelques mètres (la machine est un cube de 1,5 m de côté). L'équivalence logique entraîne la nécessité de faire "comme-si" toutes ces arêtes faisaient quelques mètres, d'où un péage lumière, corrigé par tout un tas de problèmes de discontinuité dûs aux connecteurs, etc: au moins 200 ns.

Conclusion: si la CM1 était plus petite, elle irait plus vite. Bien sûr, elle serait plus dense, mais la technologie le permet aujourd'hui.

Nécessité de paralléliser

Notre but est de faire une petite machine (une carte 30 cm x 30 cm), très puissante (de l'ordre de 1000 MIPS, 500 MFlops). La petite taille n'est pas une difficulté supplémentaire, mais au contraire la seule façon de réaliser une machine puissante.

La remarque sur la vitesse de la lumière montre également la nécessité de paralléliser. Considérons une machine classique (architecture de Von Neumann), réalisée sur une carte de 30 cm x 30 cm. Cette machine est essentiellement constituée de 2 blocs, un processeur et une mémoire, et ces blocs sont distincts. Ce qui signifie qu'en moyenne, on exécutera une instruction pour 3 transferts entre ces blocs. Vu les dimensions, et les problèmes de connectique, chaque transfert prendra de l'ordre de 10 ns, et il sera très difficile de dépasser 50 MIPS, *quel que soit l'architecture du processeur*.

Si l'on veut accélérer *cette* machine, la *seule* solution est d'attendre que l'augmentation de densité d'intégration permette de la miniaturiser davantage. Quand elle sera réalisée sur

un seul circuit (comprenant toute la mémoire), elle ira peut-être 10 fois plus vite, mais pas 100 fois. Quand elle ira 10 fois plus vite par ce gain de densité, ce gain permettra un facteur 100 par le parallélisme obtenu en utilisant la place libérée sur la même carte.

La conclusion est que notre seule chance d'atteindre des ordres de grandeur de puissance nettement supérieure, est le parallélisme, c'est à dire l'utilisation de la densité d'intégration non seulement pour réduire les dimensions de la machine, mais aussi pour multiplier les couples processeur-mémoire.

C'est d'ailleurs l'argumentation de D. Hillis [Hil85], qui a conduit à la "Connection-Machine", et nous pouvons reprendre ses arguments pour chercher quel type de parallélisme choisir:

- Combien de processeurs? Quelques gros ou beaucoup de petits (granularité)?
- Quel type de réseau d'interconnexion?
- Quel type de contrôle, SIMD ou MIMD?

A la première question, l'argumentation précédente nous conduit à répondre "beaucoup de processeurs", puisque c'est le facteur qui sera responsable de l'augmentation de puissance, donc "le maximum de petits processeurs". Mais la réponse est plus complexe, car si l'on pense pouvoir multiplier la puissance par 100, il est nécessaire de disposer de 100 processeurs, mais il n'est pas nécessaire d'aller au delà, et il est peut-être plus judicieux de faire 100 processeurs de puissance individuelle maximum que 10 000 processeurs rudimentaires.

Dans l'état actuel des choses, il est très difficile de trancher. On peut laisser guider son intuition par les 2 considérations suivantes:

- Il ne faut pas profiter du parallélisme pour augmenter la taille globale de la machine, car les communications globales auront toujours un impact sur la puissance. Ce que nous faisons en parallélisant, c'est que nous favorisons les communications à courte distance (entre processeur élémentaire et mémoire élémentaire) par rapport aux communications à grande distance. Mais il est clair que le rapport de débit moyen entre ces 2 types de communication ne peut pas être arbitrairement augmenté.

Cet aspect de taille globale de la machine n'intervient pas de la même façon dans une architecture systolique, où l'augmentation du nombre de noeuds peut ne pas pénaliser la vitesse si les communications restent à courte distance. Mais il s'agit là plutôt d'architectures spécialisées pour des algorithmes particuliers.

- Il est souhaitable de ne pas concevoir la machine avec un chiffre de puissance en tête correspondant à une date donnée, et une densité d'intégration donnée, car ces facteurs évoluent très vite par rapport aux travaux sur la parallélisation des algorithmes. A ce titre, la Connection-Machine représente un bon choix en allant vers un maximum de processeurs.

Au moment de choisir un réseau d'interconnexion entre les processeurs, nous remarquerons que la réalisation d'un hypercube sur un circuit imprimé est très simple, en tous cas énormément plus simple que dans une armoire du type de la Connection-Machine. En effet, un circuit imprimé est un empilage de couches de connexions planes (figure 7.1). L'interconnexion entre les couches est réalisée par des trous métallisés (appelés "vias"), soit traversant tout l'empilement (ce qui consomme de la surface dans toutes les couches), soit "borgnes" (traversant un minimum de couches).

Les couches sont empilées par collage à la presse. L'épaisseur totale la plus courante est 1,6 mm, assurant une rigidité mécanique suffisante pour la plupart des produits (isolant verre-époxy). On choisit l'épaisseur des isolants en fonction du nombre de couches pour obtenir la même épaisseur totale. La seule difficulté est la précision de l'alignement des couches, pour assurer que les trous percés après le collage traversent chaque couche à l'endroit prévu. Actuellement, de nombreuses PME sous-traitantes peuvent réaliser des circuits de 20 ou 30 couches, mais n'ont pas souvent l'occasion de le faire car peu de circuits électroniques demandent une telle densité de connexions.

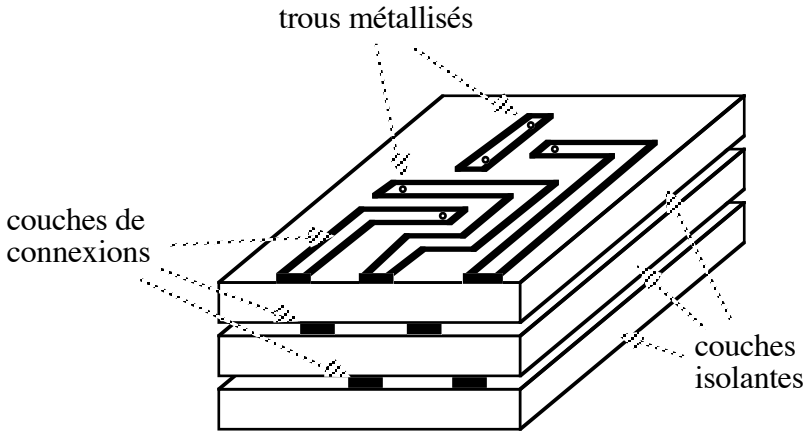

Fig. 7.1: Circuit imprimé

Un hypercube entre 2^n points se réalise par n couches de connexions, chaque couche ne comprenant que des fils parallèles entre eux (figure 7.2). Bien sûr, il faut prévoir d'autres plans de connexions pour les alimentations et les signaux de contrôle, mais disons que, en gros, un hypercube de dimension 12 doit se réaliser en une vingtaine de couches.

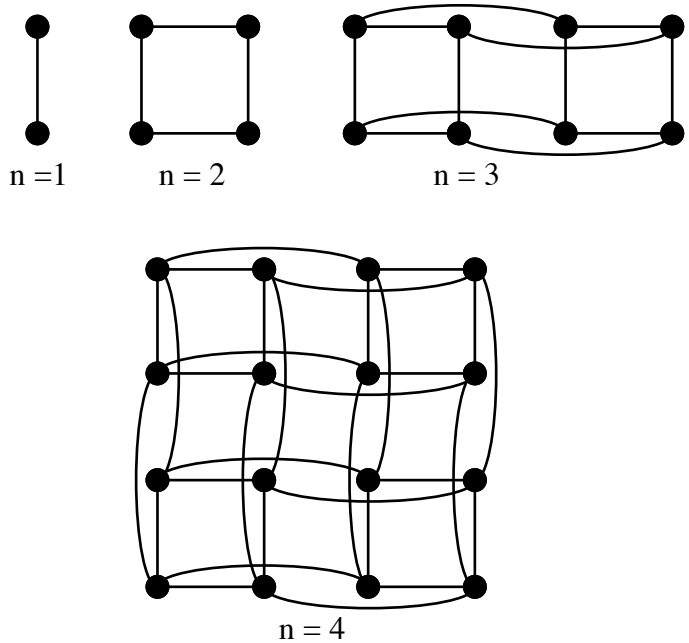

Fig. 7.2: Connexions dans un hypercube de dimension n

Après avoir considéré la granularité et le type d'interconnexion, les autres choix architecturaux nous ont semblés devoir aussi être proches de ceux de la "Connection-Machine". A la question: "SIMD ou MIMD?", nous répondons SIMD, car nous voyons mal comment on pourrait disposer quelques milliers de processeurs complexes sur une carte,

ayant chacun son séquenceur, et une UAL suffisamment sophistiquée. On pourrait répondre qu'il peut s'agir de processeurs MIMD très simples (1 bit), où l'interprétation des instructions se ferait en de nombreux cycles, mais alors, il n'y aurait pas beaucoup de différence avec une structure SIMD, sinon un point de terminologie entre ce que l'on appelle donnée et ce que l'on appelle instruction.

Quoiqu'il en soit, avant d'avancer davantage dans les choix architecturaux, nous avons décidé de nous doter d'outils pour évaluer comment se passe l'exécution d'un ou de quelques programmes type sur de telles machines, ce qui nécessite l'écriture d'un simulateur de ces machines, et pour nous guider dans la structure de ce simulateur, nous avons choisi *arbitrairement* une machine particulière, que nous allons décrire maintenant.

Structure de notre machine POMP (petit ordinateur massivement parallèle)

Nous visons une machine SIMD en une carte, c'est-à-dire avec tous les processeurs élémentaires et le micro-contrôleur sur la même carte (figure 7.3). Quand nous utilisons le mot processeur, il s'agit du couple processeur-mémoire. Pour réduire au minimum la complexité de cette première machine "*pédagogique*", nous avons décidé de commencer par un réseau d'interconnexion en grille, pour constituer une matrice de processeurs. Ceci peut paraître assez éloigné de notre but initial, mais nous avons de bonnes raisons pour nous restreindre ainsi:

- de nombreux algorithmes de traitement d'image se parallélisent bien sur ce type de machine [DL81][Duf83][JU82][Bat80][Hun81][Sa82],
- nous verrons que le schéma du processeur se prête bien à une extension en hypercube,
- la simplicité obtenue permet d'avancer vite dans l'écriture de simulateurs.

Le micro-contrôleur envoie à chaque cycle d'horloge une instruction commune à tous les processeurs. Il reçoit en entrée (par le connecteur de fond de panier) des instructions plus complexes venant de l'hôte. Le rapport de fréquence entre ces 2 débits d'instructions est lié au rapport de complexité entre les 2 types d'instructions.

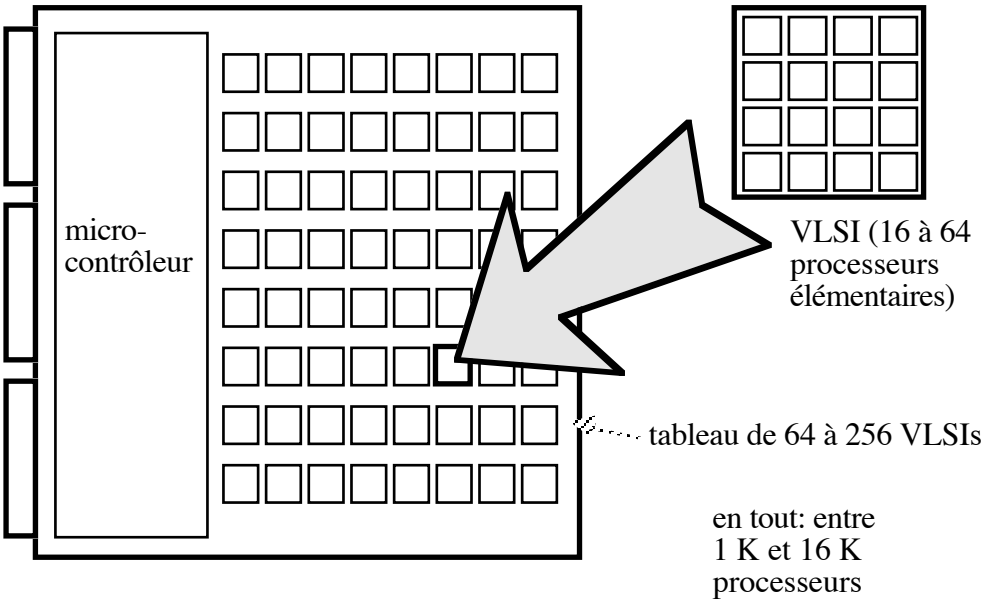

Fig. 7.3: Carte massivement parallèle

Nous visons une fréquence F de fonctionnement de 30 MHz. Si les processeurs élémentaires effectuent une addition 1 bit à cette fréquence, la puissance (maximale théorique) sera (en équivalents instructions 32 bits par seconde):

$P = F \times N / 32$ où N est le nombre de processeurs,

soit P entre 1000 et 16000 MIPS.

Mise en boîtiers des processeurs élémentaires

Nous avons dessiné (figure 7.3) sous forme d'un VLSI, une matrice de couples (processeur, mémoire). Le nombre de broches de ce boîtier sera égal au nombre de processeurs sur le périmètre, auquel il faudra ajouter l'instruction (environ 50 bits), et quelques signaux comme alimentations, horloge, etc. La densité d'intégration d'un tel VLSI sera principalement liée à la taille mémoire de chaque processeur (qui ne saurait être inférieure à 4 K bits), soit de 64 K bits à 4 M bits pour le VLSI.

Ceci nous semble trop dense pour pouvoir mener le projet rapidement. *Nous utiliserons donc des circuits mémoire du commerce*, et le VLSI ne comprendra que les processeurs (de même que dans la "Connection-Machine") (figure 7.4). Il faut donc rajouter au moins un fil par processeur pour communiquer avec sa mémoire. Si l'on met un fil par

processeur, le nombre de broches du boîtier est de la forme $b = 4\sqrt{n} + n + 50$, soit 82 broches pour 16 processeurs, et 146 broches pour 64 processeurs. Si l'on met 2 fils par processeur, $b = 4\sqrt{n} + 2n + 50$, soit 98 broches pour 16 processeurs et 210 broches pour 64 processeurs.

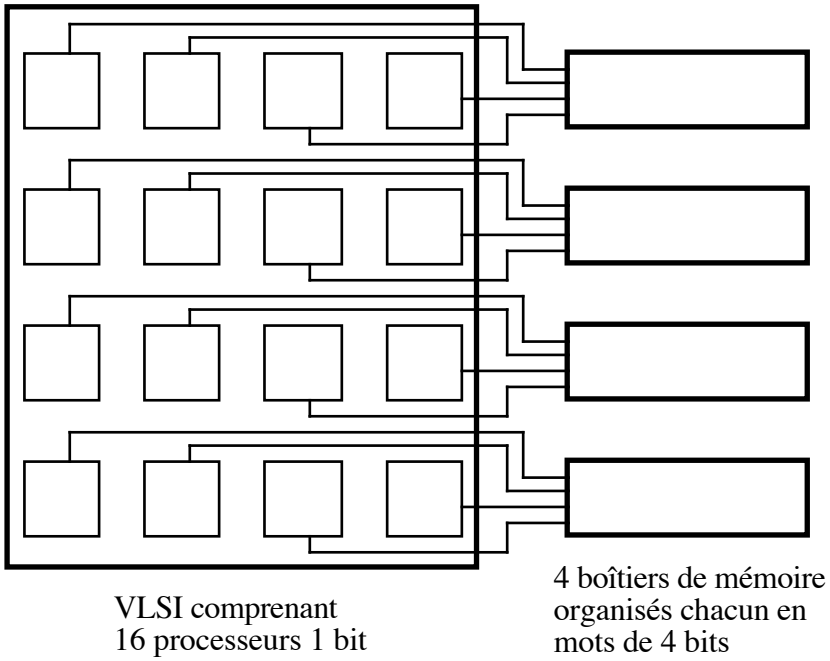

Fig. 7.4: Mémoires extérieures au circuit VLSI

Nous avons choisi une architecture de processeur qui simplifie tout cela, en mettant un fil par processeur pour communiquer avec la mémoire, et en faisant que ce même fil permette de communiquer avec les voisins (figure 7.5). Le nombre de broches est donc n + 50, ce qui permet de ne pas s'occuper pour l'instant de l'impact du nombre de processeurs par VLSI sur le nombre de broches et la complexité de ce VLSI.

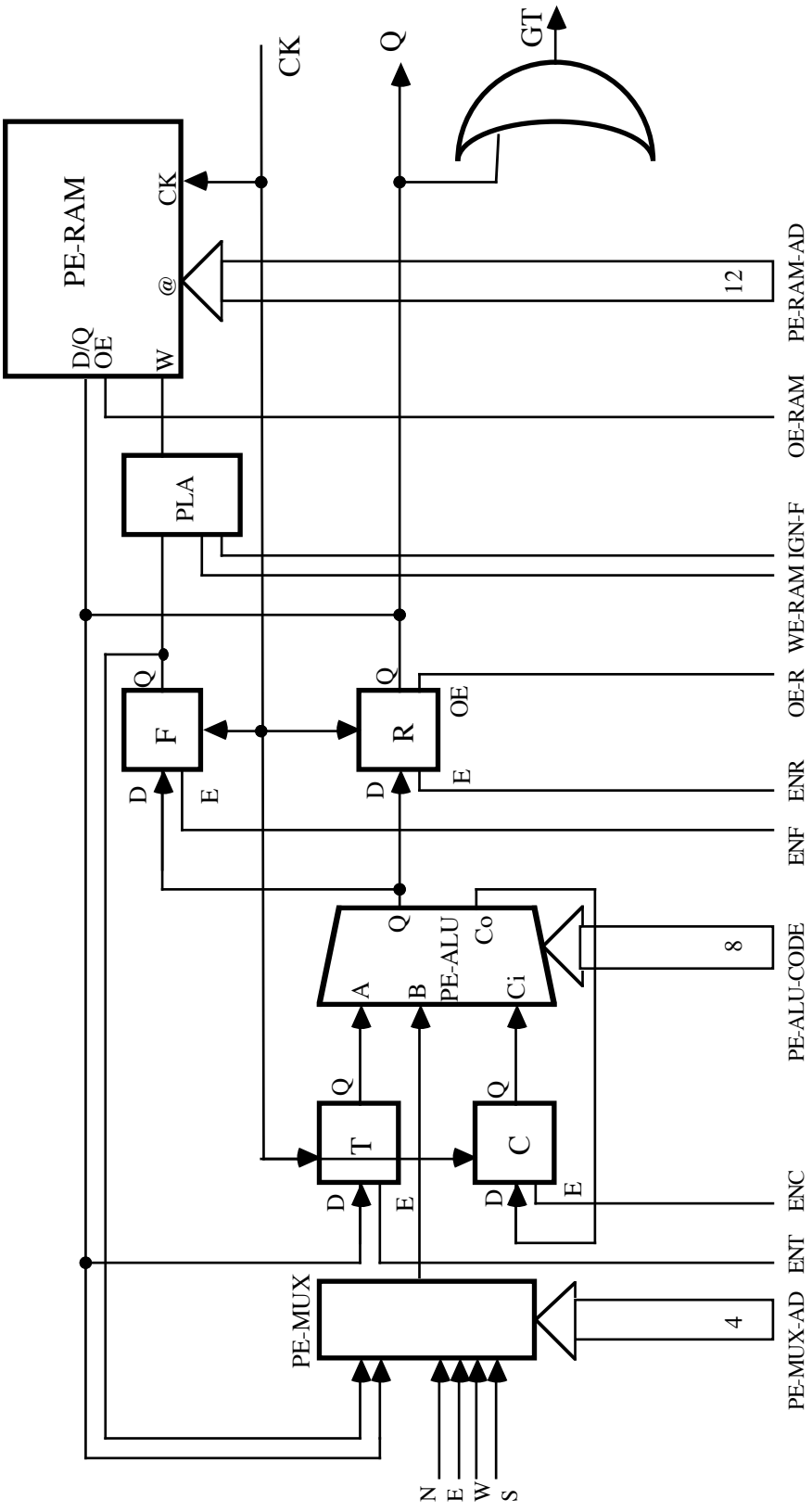

Fig. 7.5: Schéma du processeur élémentaire

Un avantage supplémentaire est que cette disposition permet tous types d'interconnexion entre les processeurs puisque les sorties de tous les processeurs sortent du circuit, et en outre, la connexion processeur - mémoire étant extérieure, il est facile

d'insérer une communication supplémentaire entre les mémoires et un autre organe extérieur (DMA avec l'hôte, etc).

Description du processeur élémentaire

T, C, R, F sont des bascules D actionnées sur le même front d'horloge (ainsi que la mémoire). Chacune reçoit un signal "Enable" issue de l'instruction.

L'UAL (PE-ALU) peut effectuer les opérations logiques et des additions avec ou sans report.

Le multiplexeur d'entrée voit les bus de ses 4 voisins (N,S,E,W) dans la version actuelle de la machine. Il sera aisé de lui ajouter des entrées.

Les adresses de la RAM (PE-RAM-AD) et du MUX (PE-MUX-AD) sont issues de l'instruction, de même que la nature de l'opération de l'UAL (PE-ALU-CODE).

F est un "flag" conditionnant l'écriture dans la RAM. En effet, il faut pouvoir rendre inactif chaque processeur individuellement, en fonction d'une condition qu'il aura lui-même calculé. La seule façon de le rendre inactif est qu'il n'écrive pas dans sa RAM. Un fil de l'instruction indique si on ignore ou non cette inhibition qui pourrait être provoquée par F (IGN-F).

Le signal GT est un OU entre toutes les sorties de tous les processeurs. C'est le seul signal qui retourne vers le micro-contrôleur, et qui permet d'arrêter une itération ou d'aiguiller un test de façon globale.

La structure du MUX permet de considérer la mémoire des voisins comme une extension de la mémoire locale, si l'on concatène l'adresse de la RAM et celle du MUX (ce qui est fait de toute façon puisque les 2 viennent de l'instruction).

L'instruction (que nous nommerons à partir de maintenant nano-instruction) comporte les bits suivants, détaillés en bas de la figure 7.5 (au nombre de 32 si la RAM fait 4 K bits):

- 12 bits d'adresse RAM,
- 4 bits d'adresse MUX,
- 8 bits de code-op de l'UAL,

- 5 bits d'Enable,
- 2 bits de contrôle d'écriture sur le bus,
- 1 bit de commande de l'utilisation de F.

Structure du micro-contrôleur

Le micro-contrôleur séquence les opérations à la fréquence de l'horloge générale (CK) et traduit les instructions venant du fond de panier (que nous nommerons macro-instructions) en suites de nano-instructions pour la matrice de processeurs.

Il reçoit des macro-instructions du genre:

- addition 32 bits de 2 variables en mémoire locale, résultat dans la mémoire locale,
- transfert d'un mot de 32 bits de la mémoire du voisin vers la mémoire locale,
- addition d'un mot de 32 bits du voisin à un mot local, résultat dans la mémoire locale,
- etc,

et traduit cela en nano-instructions du genre:

- transfert 1 bit mémoire vers T,
- addition d'un bit du voisin avec T, résultat dans R,
- transfert de R dans RAM, conditionnel à F
- etc.

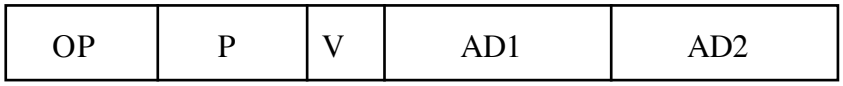

Fig. 7.6: Instruction à 2 adresses

Pour aller plus avant dans la structure du micro-contrôleur, il faut définir mieux le langage des macro-instructions. Nous avons choisi des instructions à 2 adresses du type de la figure 7.6.

- OP définit l'opération: addition, move, etc,
- P définit le facteur de répétition: n pour une addition n bits, etc,
- V définit le voisin, dans le cas d'un opérande non local,

- AD1 et AD2 sont les 2 adresses de base des opérandes p bits, c'est-à-dire l'adresse du bit de poids faible de cet opérande en mémoire.

- Le résultat sera rangé à une de ces 2 adresses, suivant la nature précise de l'instruction.

On remarquera que ce langage de macro-instructions cache complètement la structure du processeur élémentaire. Ses bascules ne sont connues que du micro-contrôleur. Ceci permettra de modifier la structure des processeurs élémentaires, et même sa largeur, sans modifier ce macro-langage.

Le travail du micro-contrôleur sera donc:

- auto-incrémenter les adresses au cours de l'opération,
- auto-décrémenter P et tester l'égalité à 0,
- choisir pour chaque nano-instruction le voisinage V ou en forcer un autre,
- et séquencer le tout.

Nous avons choisi de disposer pour ces calculs d'un UAL de la largeur du champ adresse, et contenant un certain nombre de registres de travail, par exemple à l'aide de circuits Amd 2903, 3 pour des adresses de 12 bits (figure 7.7).

Les 4 champs P, V, AD1, AD2 sont enregistrés dans un registre spécialisé au début de l'interprétation de l'instruction. Ces champs sont ensuite transférés dans les registres internes des 2903, et sont, à chaque nano-instruction, éventuellement incrémentés ou décrémentés, et une adresse est éventuellement transférée dans le registre PE-RAM-AD-REG pour le cycle suivant. On remarquera que la nano-instruction ne contenant qu'un champ adresse, il suffit d'un seul UAL pour calculer les adresses. Le prix à payer est que chaque addition 1 bit prendra en moyenne 2 cycles, mais cela est aussi lié au nombre de fils entre chaque processeur élémentaire et sa mémoire.

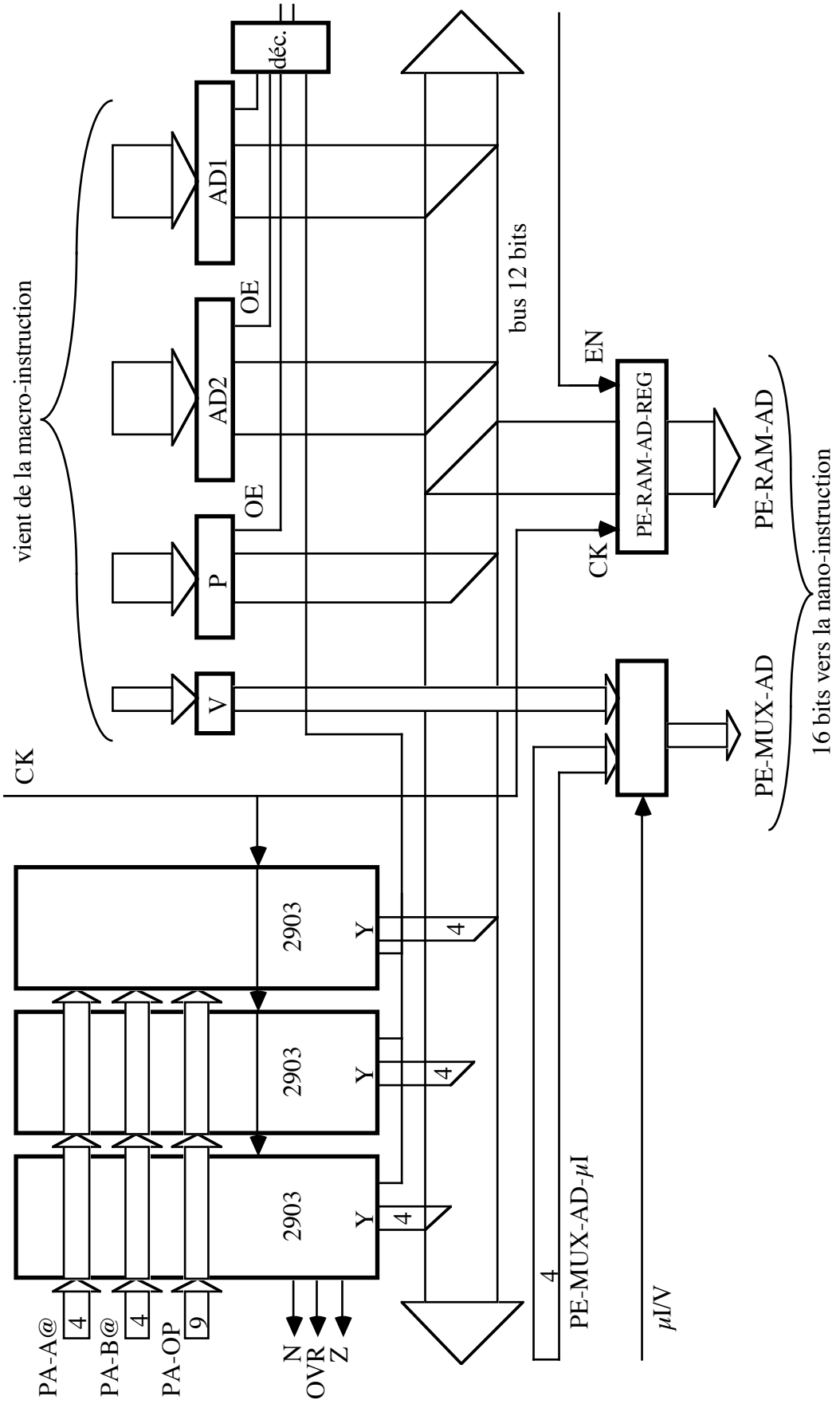

Fig. 7.7: Schéma du processeur d'adresse

Ceci pourra être modifié ultérieurement, mais seulement quand nous aurons la certitude que ceci est un facteur dominant dans l'éventuelle inefficacité de ce schéma.

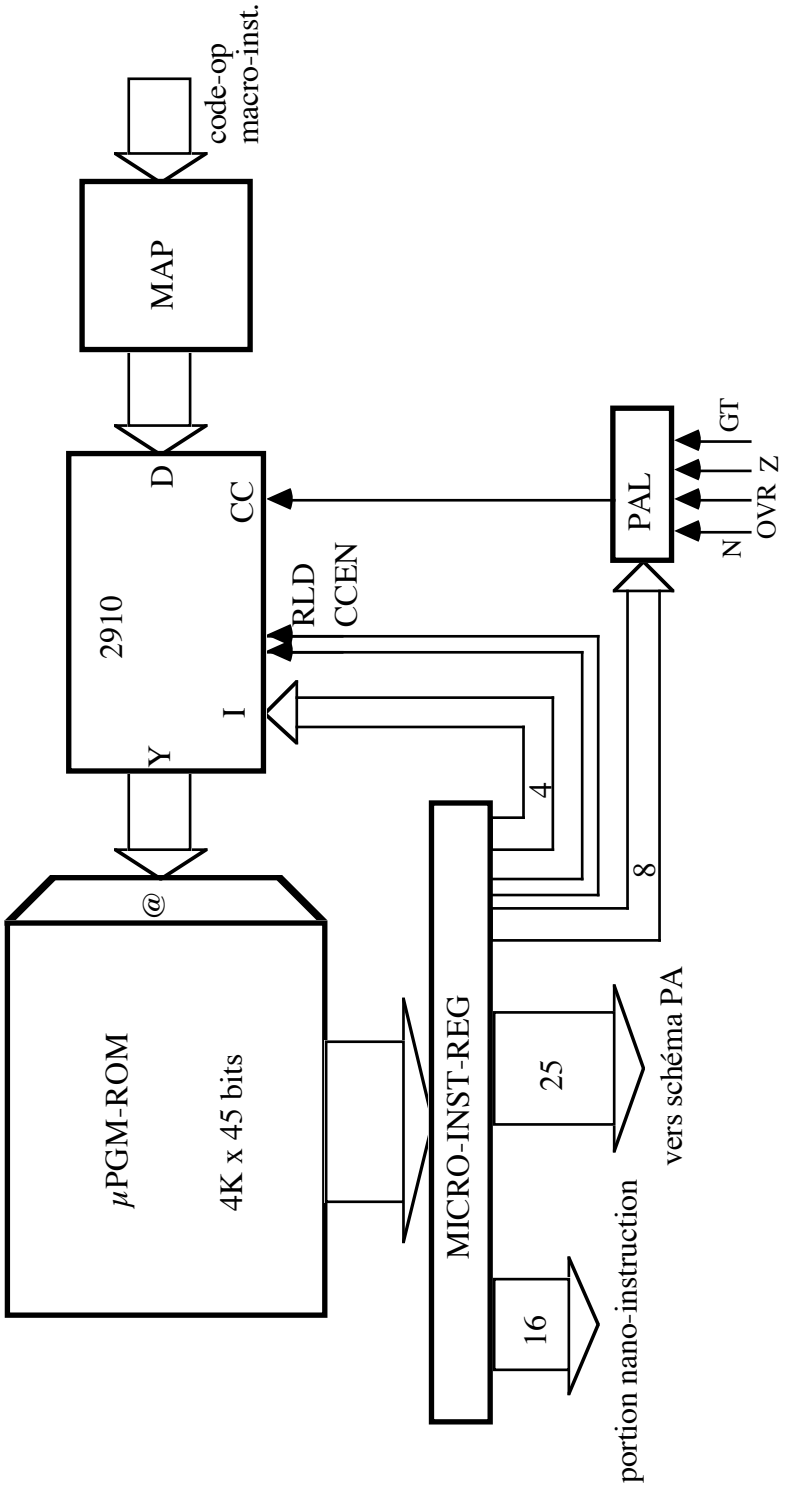

Fig. 7.8: Séquenceur du micro-contrôleur

Il ne manque plus qu'un séquenceur, qui peut être réalisé par une mémoire de micro-programme contrôlée par un Amd 2910 (figure 7.8).

Etat actuel du projet et conclusions

Eric Charpentier [Cha87] a écrit en C un simulateur de cette machine, tournant actuellement sur le VAX 11/785 du LIENS. Il est suffisamment paramétrable pour pouvoir évoluer avec le schéma logique de la machine. Il est bien entendu très lent, bien qu'optimisé, et nécessiterait d'être porté sur une machine parallèle plus puissante.

Pierre Chicourrat a écrit un assembleur de micro-programme, Théodore Papadopoulo a écrit un assembleur de macro-programme. Patrice Ossona-Demendez a définit un langage parallèle proche de C, et a écrit dans ce langage un programme de traitement d'image par recuit-simulé, inspiré par R. Azencott [Aze86].

Cette première phase du projet (3 mois) a donc consisté à commencer à mettre en place des outils. Il faut maintenant les développer et les utiliser pour définir les paramètres fondamentaux de la future machine:

- nombre de processeurs,
- taille mémoire par processeur,
- largeur en bits de chaque processeur,
- nature des opérations de l'UAL,
- nature des interconnexions entre processeurs,
- type de routage sur ces interconnexions,
- interconnexion avec l'hôte.

Ceci représente beaucoup de travail, dont une bonne partie a été faite par les équipes ayant déjà réalisé des machines massivement parallèles. Nous pensons qu'il faut s'en inspirer, sans chercher à innover, pour ce qui concerne:

- le langage de haut niveau (*C et *LISP),
- l'interconnexion avec l'hôte (langage ParIS),

mais que l'on peut développer davantage les aspects fins liés aux processeurs élémentaires et au routage, en cherchant à émuler ce qui se fait sur une machine comme la "Connection-Machine", mais de façon transparente pour l'utilisateur final.

Chapitre 8: Conclusion

Après avoir fait une analyse des problèmes posés par les architectures graphiques et le tour des solutions existantes, nous avons montré que les co-processeurs graphiques:

- ne résolvent pas (ou plus) le problème posé, qui est d'afficher des images complexes plus rapidement qu'un processeur général,
- ne doivent leur existence passagère qu'à l'insuffisance historique des vitesses des bus, des microprocesseurs et des mémoires.

Nous avons vu que les machines à générateur de vecteurs programmé et à BitBlT programmé sont plus efficaces et plus souples que les co-processeurs graphiques, et conduisent à des machines moins complexes et donc moins chères.

Nous avons réalisé et décrit trois expérimentations:

1) Carte graphique couleur 512 x 512 sur Thémis et M0,

qui explore un maximum de développements possibles à base de co-processeur, et montre la complexité de la carte obtenue sans des résultats permettant d'espérer calculer rapidement des images complexes.

2) Carte de visualisation 800 x 1000 par DMA sur M0,

qui montre que cette technique est non seulement possible pour des résolutions élevées, mais qu'elle aboutit à un montage plus simple que pour d'autres contrôleurs (disques par exemple). Elle nécessite un bus de débit élevé, mais raisonnable pour le stade actuel d'évolution des machines.

3) Machine POMP (projet en cours)

qui montre que la réalisation d'une machine massivement parallèle sous forme d'une carte additionnelle pour station de travail est possible comme opérateur d'écriture.

A partir de ces expérimentations, nous suggérons donc une solution pour la synthèse interactive d'images réalistes. Nous ne sommes pas encore en mesure de faire le schéma d'un ordinateur intégrant ces deux derniers dispositifs. Il subsiste en effet de nombreux choix à effectuer. Par exemple, comment se situera la mémoire générale de l'ordinateur hôte, par rapport à cette mémoire diluée dans la machine massivement parallèle? Il est clair que cette dernière, de par son réseau d'interconnexion interne, permet facilement un débit très élevé vers un écran. Si cette mémoire pouvait être adressée directement (avec un temps d'accès "pas trop" grand) par le processeur de la machine hôte, peut-être que cette dernière n'aurait pas besoin d'autre mémoire?

Ces questions sont simples, mais une réponse ne pourra être donnée qu'après quelques hommes x années de travail. C'est le développement d'un projet allant dans ce sens qui nous intéresse maintenant.

Chapitre 9: Références bibliographiques

[ASP*86] M. Asal, G. Short, T. Preston, R. Simpson, D. Roskell, and K. Guttag. The Texas Instruments 34010 graphics system processor. *IEEE CG&A*, 6(10), Octobre 1986.

[Aze86] Robert Azencott. Algorithmes de Recuit et Imagerie. In *Journées annuelles de la société mathématique de France*, 24-25 Janvier 1986.

[Bat80] Batcher. Design of a massively parallel processor. *IEEE Trans. on Computers*, C-29(9):836--840, 1980.

[BB80] Andreas Bechtolsheim and Forest Baskett. High-performance raster graphics for microcomputer systems. *Computer Graphics*, 14(3):43--47, July 1980.

[BBV82] Andreas Bechtolsheim, Forest Baskett, and Pratt Vaughan. *The SUN Workstation Architecture*. Technical Report-CSL-TR-229, Computer Systems Laboratory, Stanford University, Mars 1982.

[Bre65] J.-E. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25--30, July 1965.

[Bre77] J.-E. Bresenham. A linear algorithm for incremental display of digital arcs. *Comm. ACM*, 20(2):100--106, Feb 1977.

[Bru86] R. Brusq. Synthèse d'images par lancer de rayon (ray-tracing): la machine CRISTAL - résultats et perspectives. In *2ème semaine de l'image électronique*, pages-404--410, CESTA, Avril 1986.

[Cat78] Edwin Catmull. A hidden surface algorithm with antialiasing. *Computer Graphics*, 12(3):6--10, 1978.

[Cat79] Edwin Catmull. A tutorial on compensation tables. *Computer Graphics*, 13(2):1--7, August 1979.

[CB86] C. Carinalli and J. Blair. National's advanced graphics chip set for high-performance graphics. *IEEE CG&A*, 6(10), Oct 1986.

[CH80a] J.-H. Clark and M.-R. Hannah. A high performance smart image memory. *Lambda*, 1(3), 1980.

[CH80b] James-H. Clark and Marc-R. Hannah. Distributed processing in a high-performance smart image memory. *Lambda*, 1(4), 1980.

[Cha87] Eric Charpentier. 1er Septembre 1987. Rapport de stage de DEA, Université d'Orléans.

[CL76] R. Carrasco et J. Lauret. *Cours Fondamental de Télévision, Emission-Réception*. Editions Radio, 1976.

[Cla80a] James Clark. A VLSI geometry processor for graphics. *Computer*, 59--69, July 1980.

[Cla80b] James-H. Clark. Structuring a VLSI system architecture. Lambda, 1(2), 1980.

[Cla82] J.-H. Clark. The geometry engine: a VLSI geometry system for graphics. *Computer Graphics*, 16(3):127--133, July 1982.

[CLP81] D.-W. Clark, B.-W. Lampson, and K.-A. Pier. The memory system of a high-performance personal computer. *IEEE Transactions on Computers*, C-30(10):715--733, Oct 1981. 1ère publication: rapport XEROX CSL-81-1 Janv 81.

[Con87] Margery Conner. Graphics engines. *EDN*, p112, 4 Mars 1987.

[Cro77] Franklin-C. Crow. The aliasing problem in computer generated shaded images. *Comm. ACM*, 20(11):799--805, Nov. 1977.

[Cro78] Franklin C. Crow. The use of grayscale for improved raster display of vectors and characters. *Computer Graphics*, 12(2), 1978.

[Cro88] Franklin C. Crow. 3D Image Synthesis On The Connection Machine. *International Conference and Exhibition on Parallel Processing for Computer Vision and Display*, 12-15 Janvier 1988.

[Cue84] J.-C. Cuénod. Bus de M0 - Version 1.1. Octobre 1984. Rapport interne, LIE.

[DDPP86] C. Douady, D. Dure, N. Paris, et A. Permuy. FLIP: 2 architectures de multi-fenêtrage graphique. Mai 1986. Projet soumis au Concours Cray 1986, ayant obtenu le prix spécial du jury.

[Den73] E. Denert. A method for computing points of a circle using only integers. *Computer Graphics & Image Processing*, 2:83, 1973.

[DL81] M.-J.-B. Duff and S. Levialdi, editors. *Languages and Architectures for Image Processing*. Academic Press, 1981.

[Dou84] César Douady. Etude d'un chemin de donnée pour une visu graphique haute performance. Sept 1984. Rapport de stage de DEA, Université de Paris XI (Orsay).

[Duf83] M.-J.-B. Duff, editor. *Computing Structures for Image Processing*. Academic Press, 1983.

[Fer81] F.-Nunes Ferreira. Conception et réalisation d'un système interactif pour la synthèse d'images réalistes: HELIOS. Sept 1981. Thèse de Docteur-Ingénieur.

[FGH*85] H.-J. Fuchs, J.-P. Goldfeather, S. Hultquist, J. Spach, F.-P. Austin, J. Brooks-Jr, J. Eyles, and J. Poulton. Fast spheres, textures, transparencies, and image enhancements in pixel-planes. *Computer Graphics*, 19(3):111--120, Août 1985.

[FP80] Henry Fuchs and John Poulton. Pixel planes: a cellular VLSI-oriented design for a raster graphics engine. Nov. 26, 1980.

[FP81] Henry Fuchs and John Poulton. Pixel-planes: a VLSI-oriented design for a raster graphics engine. *VLSI Design*, 2(3), 1981.

[FPEG88] Henry Fuchs, John Poulton, John Eyles, and Trey Greer. Coarse-Grain and Fine-Grain Parallelism in the Next Generation Pixel-planes Graphics System. *International Conference and Exhibition on Parallel Processing for Computer Vision and Display*, 12-15 Janvier 1988.

[FPPB82] Henry Fuchs, John Poulton, Alan Paeth, and Alan Bell. Developing pixel-planes, a smart memory-based raster graphics system. In *Conference on Advanced Research in VLSI, Cambridge Mass.*, Janv, 25-27 1982.

[Fre86] K. Frenkel. Evaluating two massively parallel machines. *Comm. ACM*, 29(8):752--758, Août 1986.

[Gas77] Jean Gastinel. Conception et intégration d'un terminal alphanumérique. 20 Janv. 1977. Thèse de 3ème cycle, Université P. et M. Curie, Paris.

[GB77] J. Gastinel and A. Bernardy. The design of a low-cost tty replacement. In *Euromicro 77*, 1977.

[GF86] J. Goldfeather and H. Fuchs. Quadratic surface rendering on a logic-enhanced frame-buffer memory. *IEEE CG&A*, 48--59, Jan 1986.

[GHF86] Jack Goldfeather, Jeff-P.-M. Hultquist, and Henry Fuchs. Fast constructive-solid geometry display in the pixel-powers graphics system. *Computer Graphics*, 20(4):107--116, Août 1986.

[Hil85] W.-Daniel Hillis. *The Connection Machine*. MIT Press, 1985.

[Hor76] Berthold-K.-P. Horn. Circle generators for display devices. *Computer Graphics and Image Processing*, 5:280--288, 1976.

[Hun81] D.-J. Hunt. The ICL DAP and its application to image processing. In M.-J.-B. Duff and S. Levialdi, editors, *Languages and Architectures for Image Processing*, Academic Press, 1981.

[Ing81] D.-H. Ingalls. The smalltalk graphics kernel. *Byte*, 6:168--194, Aout 1981.

[JG78] G.-H. Joblove and D. Greenberg. Color spaces for computer graphics. *Computer Graphics*, 1978.

[JU82] Preston Jr. and L. Uhr, editors. *Multicomputers and Image Processing*. Academic Press, N. Y., 1982.

[Kay77] Alan Kay. Microelectronics and the personal computer. *Scientific American*, 237(3):230--244, Sept 1977.

[KR75] C.-V. Kameswara-Rao. Comment on a method for computing points of a circle using only integers. *Computer Graphics & Image Processing*, 4:79, 1975.

[Kri84] Suresh Krishna. Etude et réalisation du contrôle et de l'interfaçage d'une visu graphique haute performance. 1984. Rapport de stage de DEA, Université de Paris XI (Orsay).

[KU81] J. Kajiya and M. Ullner. Filtering high quality text for display on raster scan devices. *Computer Graphics*, 15(3), 1981.

[Ler80] P. Leray. Réalisation d'un système de génération synthétique d'images en temps réel. In *Afcet III, Nancy*, nov 1980.

[Ler83] P. Leray. Conception d'un système de synthèse d'images tridimensionnelles animées pour la production télématique et audiovisuelle. *Radiodiffusion-Télévision*, (80):4--7, 1983.

[Lie86] Bruno Liège. Conception et réalisation d'une visu Bitmap 100 MHz avec accès direct à la mémoire. 21 Juin 1986. Rapport de DEA, Université Paris XI (Orsay).

[LP80] B.-W. Lampson and K.-A. Pier. A processor for a high-performance personal computer. In *7th Int. Symp. Computer Architec., Sigarch, IEEE, La Baule*, pages-146--160, May 1980. Republié en rapport XEROX CSL-81-1.

[Mar84] F. Martinez. Vers une architecture banalisée des synthétiseurs d'image. In *1ère semaine internationale de l'image, Biarritz*, 1984.

[Mar86] Francis Martinez. Architectures spécialisées et technologie en synthèse d'image. In *Semaine internationale de l'image électronique, 2ème colloque image, Nice*, pages-353--359, CESTA, Avril 1986.

[Mat78a] Philippe Matherat. A chip for low-cost raster-scan graphic display. *Computer Graphics*, 12(3):181, Août 1978.

[Mat78b] Philippe Matherat. Conception d'un circuit intégré pour la visualisation graphique. 19 Mai 1978. Thèse de 3ème cycle, Université P. et M. Curie, Paris.

[Mat80] Philippe Matherat. Le circuit EF9365, notice de description technique. 1980. notice EFCIS, référence: SP01-F.

[Mat84] P. Matherat. Vers un contrôleur d'écran graphique VLSI. *TSI*, 3(2):129--141, 1984.

[MBF*80] P. Matherat, D. Bouteaud, N. Forget, J. Lebrun, et J.-P. Moreau. A high-performance integrated true graphic processor. In *ESSCIRC 80*, Grenoble, 1980.

[MF81] Francis Martinez and Fernando Ferreira. HELIOS: terminal interactif pour la synthèse d'images réalistes. In *Congrès Afcet*, pages-489--501, Nov, 18-20 1981.

[MS68] T.-H. Myer and I.-E. Sutherland. On the design of display processors. *CACM*, 11(6):410, Juin 1968.

[NS79] William-M. Newman and Robert-F. Sproull. *Principles of Interactive Computer Graphics*. McGraw-Hill, second edition, 1979. Première édition en 73.

[OHU*81] Tetsuji Oguchi, Misao Higuchi, Takashi Uno, Michinori Kamaya, and Munekazu Suzuki. A single-chip graphic display controller. In *Digest of Technical Papers*, page-170, IEEE International Solid-State Circuits Conference, 1981. Description du chip NEC 7220.

[Par86] N. Paris. FLIP: un circuit intégré pour chemin de données graphiques. 25 Juin 1986. Rapport de stage de DEA, Université Paris XI (Orsay).

[PFA*85] J. Poulton, H. Fuchs, J.-D. Austin, J.-G. Eyles, J. Heinecke, C. Hsieh, J. Goldfeather, J.-P. Hultquist, and S. Spach. Pixel-planes: building a VLSI based raster graphics system. In *Chapell Hill Conference on VLSI*, pages-35--60, 1985.

[PGI84] Rob Pike, Leo Guibas, and Dan Ingalls. Bitmap graphics. 1984. Siggraph'84 Course Notes.

[Pie83] Kenneth-A. Pier. *A retrospective on the Dorado, a high performance personal computer*. Technical Report-XEROX ISL-83-1, XEROX-Parc, Aout 1983.

[Pik84] R. Pike. The Blit: a multiplexed graphics terminal. *AT&T Bell Laboratories Technical Journal*, 63(8), Oct 1984.

[Pit67] M.-L.-V. Pitteway. Algorithm for drawing ellipses and hyperbolae with a digital plotter. *Computer*, 10:282--289, 1967.

[PLR85] Rob Pike, Bart Locanthi, and John Reisert. Hardware/software tradeoffs for bitmap graphics on the blit. *Software Practice and Experience*, 15, Mars 1985.

[Rus78] R.-M. Russell. The Cray-1 computer system. *Comm. ACM*, 21(1):63--72, Jan 1978.

[Sa82] D.-H. Schaeffer and al. The massively parallel processor. *Engineering Notes*, 5(3):313--315, 1982.

[Sch81] B.-J. Schachter. Computer image generation for flight simulation. *IEEE CG&A*, 1(4):29--68, Oct 1981.

[Sei85] C. Seitz. The cosmic cube. *Comm. ACM,* 28(1):22--33, Jan 1985.

[SH74] I.-E. Sutherland and G.-N. Hodgman. Reentrant polygon clipping. *Comm. ACM*, 17(1):32, 1974.

[Shi86] G. Shires. A new Vlsi graphics coprocessor - the Intel 82786. *IEEE CG&A*, 6(10), Oct 1986.

[Sho79] Richard-G. Shoup. Color table animation. *Computer Graphics*, 13(2):8, 1979.

[SII*85] H. Sato, H. Ishihata, M. Ishii, M. Kashimoto, K. Sato, K. Hirota, M. Ikesaka, and K. Inoue. Fast image generation of constructive solid geometry using a cellular array processor. *Computer Graphics*, 19(3):95--102, 1985.

[SS68] R.-F. Sproull and I.-E. Sutherland. A clipping divider. In Thompson Books, editor, *AFIPS FJCC 1968*, pages-765--776, Washington D.C., 1968.

[SSS74] Ivan-E. Sutherland, Robert-F. Sproull, and Robert-A. Schumaker. A characterization of ten hidden-surface algorithms. *Computing Surveys*, 6(1):1--55, March 1974.

[Sut68] I.-E. Sutherland. A head mounted three dimensional display. In Thompson Books, editor, *FJCC 1968*, page-757, Washington D.C., 1968.

[TML*82] C.-P. Thacker, E.-M. McCreight, B.-W. Lampson, R.-F. Sproull, and D.-R. Boggs. Alto: a personal computer. In Daniel-P. Siewiorek, C.-Gordon Bell, and Allen Newell, editors, *Computer structures: Principles and examples*, pages-549--572, McGraw-Hill, 1982. Première publication: rapport XEROX CSL-79-11, August 7, 1979.

[VPP*86] Pierre Vincens, Nicolas Paris, Jean-Luc Pujol, Christine Gaboriaud, Thierry Rabillaud, Jean-Louis Pennetier, Philippe Matherat, and Philippe Tarroux. HERMeS: A second Generation Approach to the Automatic Analysis of Two-Dimensional Electrophoresis Gels; Part I: Data Acquisition. *Electrophoresis*, (7):347--356, 1986.

[War80] John-E. Warnock. The display of characters using gray level sample arrays. *Computer Graphics*, 14(3):302--307, July 1980.

[Wei81] Richard Weinberg. Parallel processing image synthesis and anti-aliasing. *Computer Graphics*, 15(3):55--62, 1981.

[Wil84] Greg Williams. The Apple Macintosh computer. *Byte*, 9(2):30, February 1984.

Glossaire

3D

A trois dimensions. Adjectif. Se dit pour des images. En fait, les images sont toujours planes, donc à 2 dimensions. Quand on utilise ce terme, on veut indiquer que la base de données décrivant les objets représentés sur l'image comprend les coordonnées dans l'espace à 3 dimensions. Ce qui signifie que l'on calcule l'image par une perspective, et qu'un changement de point de vue n'affecte pas la base de données.

Allocateur

Circuiterie réalisant une allocation, en général d'un bus modulaire ou d'une mémoire, à un dispositif (par exemple un processeur) ayant positionné un signal de requête, dans le but d'en devenir maître pour réaliser un échange (un cycle d'accès à la mémoire ou un transfert élémentaire sur le bus).

Amd

American Micro Devices. Nom d'une société américaine commercialisant des circuits intégrés. Les circuits de la famille 2900 (2900, 2901, 2903, 2910, etc) permettent de réaliser des unités centrales microprogrammées. Ils permettent à peu près toutes les structures imaginables, et sont universellement employés dans ce but. Il y a de nombreuses "secondes sources" (circuits équivalents commercialisés sous licence par d'autres sociétés), mais pas de concurrents. Ils constituent une "norme de fait".

ASCII

American Standard Codes for Information Interexchange. Nom de la norme de codage de caractères alphanumériques la plus répandue. C'est le codage utilisé pour les caractères sur une ligne de transmission entre un ordinateur et un terminal. Outre les caractères, sont codés des symboles de mise en page et de contrôle du terminal. La norme pré-

voie l'extension à d'autres symboles par utilisation du caractère "Escape" qui intervient comme modifieur du ou des caractères suivants.

AsGa

Symbole chimique de l'Arséniure de Galium. Désigne une technologie de réalisation de circuits intégrés, dans laquelle ce composé remplace le Silicium. C'est la technologie qui permet les temps de commutation les plus courts à la température ambiante.

Barrel-shifter

Décaleur "à barillet". Circuiterie permettant d'effectuer une permutation circulaire de n bits sur les b bits d'un mot, en un cycle d'horloge et non en n comme dans un registre à décalage classique.

BitBlT

Bit Block Transfert. Fonction généralisée de copie de rectangles de pixels avec opération logique pixel à pixel et clipping, utilisée sur les écrans "bit-map". Cette fonction, inventée au Laboratoire de Xerox-PARC dans les années 70, permet d'unifier l'écriture de caractères alphanumériques, la gestion de fenêtres, de menus, et le défilement du contenu des fenêtres ("scroll").

Bit-map

Façon de gérer un écran monochrome. Si l'on ne se réfère qu'à une traduction mot à mot, il s'agirait d'une méthode où l'on code en mémoire un tableau de bits représentant les pixels (chacun sur un bit) de l'écran. Or ceci est le cas de toutes les visus à balayage télévision. Ce terme est apparu avec les concepts développés à Xerox-PARC. Il intègre en fait toute la philosophie des machines et des logiciels Alto et Dorado: la mémoire d'image est une zone banalisée de la mémoire générale, les opérations d'écriture ne font appel à aucune circuiterie spécialisée, la primitive est le BitBlT.

Buffer

Tampon mémoire. Utilisé de façon un peu particulière pour les mémoires d'image, en particulier dans l'expression "double-buffering", où il s'agit en général de mémoire spécialisée, liée aux signaux de contrôle du moniteur télévision.

CCD

Charge Coupled Devices. Dispositifs à charges couplées. Circuits intégrés dans lesquels l'utilisation systématique de très grands registres à décalages rudimentaires (chaînes de condensateurs reliés par des transistors) permet de grandes densités d'intégration et de

mémorisation. Si les points élémentaires sont photosensibles, on a alors un capteur d'image dans lequel la sérialisation pour obtenir le signal vidéo est déjà réalisée. C'est la technologie qui se répand de plus en plus pour la réalisation "solid-state" de caméras miniaturisées et très légères. Dans ces caméras, l'image acquise est échantillonnée sur les points d'intersection d'une grille, alors que dans une caméra vidéo classique, le signal décrivant l'image est continu le long d'une ligne.

Clipping

Coupage? ou coupure? des objets sur un bord de fenêtre ou d'écran. Opération indispensable avant le dessin de l'objet en mémoire d'image pour éviter les débordements des pointeurs manipulés par les primitives de tracé.

CMOS

Complementary Metal Oxyde Semiconductor. Technologie de circuits intégrés MOS dans laquelle on utilise conjointement des transistors de type N et des transistors de type P dans les portes élémentaires. Le principal intérêt est de rendre la consommation quasi-nulle en dehors des instants de commutation. C'est la technologie qui a rendu possible les montres électroniques et tous les objets alimentés par une pile minuscule. Actuellement, cette technologie permet de très grandes densités (plusieurs millions de transistors par circuits) et des temps de commutation très courts (inférieurs à la nanoseconde à l'intérieur d'un circuit), rendant moins intéressantes les tehnologies considérées comme plus rapides (comme l'ECL dont la vitesse n'est plus que de l'ordre de 2 fois plus grande).

Code-op

Code d'une opération élémentaire dans une machine. Désigne uniquement le champ de l'instruction qui code la nature de l'opération, et non les champs qui codent les adresses des opérandes.

Crossbar

Circuiterie qui permet de connecter par continuité électrique simultannée un ensemble de fils à un autre ensemble de fils. C'est par exemple ce qui est réalisé par un commutateur de standard téléphonique classique.

CSG

Constructive Solid Geometry. Technique de codage de base de données d'objets pour la synthèse d'image. Dans cette technique, les objets sont définis par une arborescence d'opérations ensemblistes (union, intersection) sur des formes géométriques simples (que l'on peut définir par des équations paramétriques simples). Cette technique est utilisée en

particulier en CAO de pièces mécaniques où les objets sont souvents des combinaisons de formes simples réalisables par les machines-outils (arêtes droites, faces planes, cylindres, cônes, etc).

DAC

Digital / Analog Converter. Convertisseur digital / analogique, ou plutôt en français normalisé: convertisseur numérique / analogique (CNA). Malheureusement, ces formes sont trop mal choisies et trop peu utilisées pour être compréhensibles. C'est surtout le "numérique" qui ne veut rien dire, car tout est numérique, aussi bien à l'entrée qu'à la sortie.

DMA

Direct Memory Access. Accès direct à la mémoire. Quand on utilise ce terme, on pense à un type d'accès précis à la mémoire: celui d'une circuiterie qui émet des adresses vers la mémoire, et qui comprend généralement sur le chemin de données un FIFO pour résoudre l'asynchronisme des horloges, celle de la mémoire d'une part, et celle du dispositif ayant besoin de cet accès d'autre part. C'est donc un dispositif qui court-circuite le processeur central pour obtenir un débit plus important que si cette opération était programmée. Ce sigle anglais utilisé en français est donc davantage porteur de sens qu'il ne doit l'être pour les ingénieurs de langue anglaise. C'est un des avantages de l'intégration de termes étrangers.

ECL

Emitter Coupled Logic. Technologie de réalisation de portes logiques à base de transistors bipolaires, où les sorties des portes sont réalisées par les émetteurs des transistors. Cette technologie, évitant en outre la saturation des transistors, permet des temps de commutation très courts, et est utilisée dans les supercalculateurs (CRAY). Malheureusement, la consommation électrique (donc la dissipation thermique) est grande, et la densité d'intégration ne peut pas être élevée (quelques portes par circuit).

Enable

Signal d'autorisation de commutation. En général, signal d'entrée d'un circuit associé à une entrée d'horloge, indiquant au circuit s'il est concerné ou non par la commutation sur le front de l'horloge.

Escape

Code d'un symbole du code ASCII permettant d'étendre les fonctions de celui-ci.

Ethernet

Nom d'une norme de communication entre ordinateurs par un câble coaxial. L'essentiel de cette technique a été développée au Laboratoire de Xerox-PARC, par l'équipe ayant inventé les ordinateurs individuels et la gestion des écrans "bit-map".

Fetch

Opération effectuée par un processeur et consistant à aller chercher en mémoire l'instruction qui suit celle qu'il vient d'exécuter. Substantif associé au verbe "to fetch". A l'avantage d'être très court et très précis en français si l'on n'utilise que ce sens là du mot anglais, qui est certainement plus vague pour un anglophone.

FIFO

First In, First Out. Pile dans laquelle le premier entré est le premier sorti. En français: queue. Quand on utilise ce terme dans un contexte d'architecture des ordinateurs, il a un sens plus précis qu'une simple queue qui pourrait être implémentée de diverses façons. Il s'agit toujours d'un circuit qui effectue internement des décalages asynchrones, car les horloges d'entrée et de sortie n'ont pas de rapport de phase (ne sont pas issues d'une même horloge-mère par division de fréquence). Ces circuits sont indispensables dans les contrôleurs de périphériques où l'on n'a pas le choix de l'horloge: disque, ligne de communication.

Flag

Drapeau ou sémaphore. Très court et très précis, car désigne toujours une variable sur un seul bit. Encore un mot qui a davantage de sens en français qu'en anglais.

GFlops

Giga Floating point Operation Per Second. Milliard d'opérations arithmétiques sur des nombres à virgule flottante par seconde.

GKS

Graphic Kernel Standard. Nom d'une norme définissant les appels et les fonctions des routines de bibliothèques graphiques.

HF

Haute Fréquence. Fréquence des ondes électromagnétiques utilisées pour les transmissions radio.

Josephson

Nom du physicien qui a découvert un effet (qui porte son nom) qui se manifeste lorsqu'un corps devient supraconducteur (à très basse température - quelques Kelvin). En utilisant cet effet, on peut réaliser des circuits intégrés dont les portes commutent en quelques pico-secondes. Durant de nombreuses années, les recherches dans ce domaine n'ont pu déboucher sur des produits commerciaux, à cause du prix de l'Hélium et de sa liquéfaction, nécessaires pour obtenir les températures exigées. Tout récemment ont été découverts des composés supraconducteurs à des températures beaucoup plus élevées (au moins celle de liquéfaction de l'Azote, un des corps les plus répandus, se liquéfiant à l'aide d'une pompe à chaleur guère plus complexe que celle d'un réfrigérateur). Il reste à mettre au point des circuits électroniques denses avec ces nouveaux composés, ce qui peut demander un délai important.

LSI

Large Scale Integration. Intégration à grande échelle. Désigne les circuits intégrés comportant un ordre de grandeur de quelques milliers de transistors.

MFlops

Mega Floating point Operations Per Second. Million d'opérations arithmétiques sur des nombres à virgule flottante par seconde.

MIMD

Multiple Instructions Multiple Datas. Désigne la classe des machines parallèles dans lesquelles chaque processeur exécute un flot d'instructions particulier, par opposition à SIMD.

MIPS

Mega Instructions Per Second. Million d'instructions par secondes. Ne concerne que les opérations sur des nombres entiers ou à virgule fixe, et généralement de l'ordre de complexité d'une addition sur 32 bits.

Move

Instruction élémentaire de déplacement d'une donnée dans un ordinateur.

MSI

Middle Scale Integration. Intégration à moyenne échelle. Désigne les circuits intégrés comprenant un nombre de l'ordre de quelques centaines de transistors.

Multibus

Norme de bus de communication entre cartes introduite par la société Intel.

MUX

Abréviation de "multiplexer", en français multiplexeur. Désigne une circuiterie réalisant un aiguillage, en positionnant sur sa sortie la copie de celle de ses n entrées qui est désignée par le code introduit sur des entrées spéciales d'adresse. Ce terme vraiment très court a deux avantages: il se prononce, et il ne prend pas beaucoup de place sur les schémas.

Overhead

Désigne la pénalisation à laquelle on n'a pas pensé à temps, et qu'il faut se résigner à ajouter aux prédictions de performances optimistes. On pourrait traduire par TVA?

ParIS

Parallel Instruction Set. Jeu d'instructions défini pour la "Connection Machine", permettant d'effectuer en parallèle des opérations identiques sur un grand nombre de données différentes, et adapté à cette architecture SIMD.

Pipe-line

Métaphore désignant une technique d'augmentation de performances utilisée pour les ordinateurs et qui consiste à découper les circuits en suites de circuits plus simples séparés par des registres mémorisant les résultats intermédiaires. On augmente ainsi le temps de transfert total, mais on fractionne le temps élémentaire d'introduction des opérandes dans le circuit. Le débit d'opération est alors augmenté, dans le cas où l'on doit effectuer une même opération sur de nombreux opérandes successifs (utilisé largement dans les machines "vectorielles").

PLA

Programmable Logic Array. Circuit réalisant une fonction logique programmable, en général une fonction combinatoire assez simple, et que l'on peut programmer de diverses façons, par exemple en grillant des fusibles grâce à un appareil spécial.

Pop-up menu

Dans la philosophie des écrans "bit-map", menus qui apparaissent quand on "clicke" avec la souris, et qui permettent de lancer une commande sans avoir ni à la taper au clavier, ni à chercher dans la documentation égarée la liste des commandes disponibles, avec leur orthographe et l'ordre de spécification des options.

PROM

Programmable Read Only Memory. Mémoire à lecture seulement, programmable. Ce qui est déjà un contresens. ROM désigne des mémoires dont le contenu est défini à la fabrication. Programmable signifie qu'on doit les programmer à l'installation, par un mécanisme irréversible, comme le claquage éventuel d'un fusible par bit. Il existe des mécanismes de programmation réversibles: les UVPROM se programment électriquement et s'effacent par l'exposition à des rayons ultraviolets; les EEPROM se programment et s'effacent électriquement (EE signifie Electrically Erasable). Ce sigle signifie donc 2 choses: d'une part l'écriture est lente (quelques millisecondes), et d'autre part, ces mémoires ne sont pas "volatiles", c'est-à-dire qu'elles ne perdent pas leur contenu lorsqu'on supprime l'alimentation, contrairement aux RAM.

RAM

Random Access Memory. Mémoire à accès aléatoire. Par opposition à accès séquentiel, comme les disques. Ce sigle n'est pas intéressant par sa signification mot à mot, mais par l'opposition RAM / ROM. Les ROM sont aussi à accès aléatoire. Par contre les RAM peuvent être écrites aussi rapidement que lues.

RISC

Reduced Instruction Set Computer. Ordinateur à jeu d'instructions réduit. En fait, dans ces ordinateurs, non seulement le jeu est réduit, mais en outre, les instructions sont simples. Philosophie de conception d'ordinateurs dans laquelle on pense que la simplification permet l'accélération des circuits, et qui est basée sur des mesures statistiques montrant que les instructions complexes sont peu utilisées, alors qu'elles ralentissent toutes les autres par la circuiterie qu'elles obligent à rajouter.

RMW

Read Modify Write. Lecture modification écriture. Cycle d'accès à une mémoire qui permet de lire puis d'écrire la même case mémoire en une seule opération. Ce type de cycle est possible sur les mémoires dynamiques où il prend moins de temps que deux cycles successifs car il n'y a qu'une adresse à décoder.

Roll-up

Autre terme pour "scroll" vertical vers le haut, répandu avant le succès des écrans "bit-map". Utilisé sur les écrans uniquement alphanumériques, où c'était le seul "scroll" possible.

Scroll

Petit déplacement vertical ou horizontal de tous les pixels d'une fenêtre sur un écran.

SIMD

Single Instruction, Multiple Datas. Instruction unique, données multiples. Désigne la classe des machines parallèles dans lesquelles tous les processeurs exécutent simultanément la même instruction, chacun sur des données particulières. Par opposition à MIMD.

Slot

Emplacement d'une carte additionnelle dans un ordinateur à bus modulaire. Allusion aux fentes des machines à sous ("slot machines").

THT

Très Haute Tension. Désigne la tension finale d'accélération des électrons dans un tube à rayons cathodiques. Sa valeur est de l'ordre de 17 kV. Elle doit être élevée pour obtenir une bonne luminosité, mais une valeur trop élevée entraîne une trop forte émission de rayons X par l'écran. Des normes définissent donc un maximum pour cette tension.

UAL

Unité Arithmétique et Logique. La partie du processeur qui effectue les opérations sur les nombres manipulés. Appelée quelquefois "boîte à opérations".

VLSI

Very Large Scale Integration. Intégration à très grande échelle. Désigne les circuits intégrés comprenant quelques dizaines à quelques centaines de miliers de transistors.

VME

Norme de bus modulaire de connexion entre cartes, introduite par la société Motorola pour le marché européen, et ayant eu un succès mondial.

Wrapping

Technique d'interconnexion par enroulement de fils de cuivre argentés sur des pattes dorées de section carrée, de très haute fiabilité et de modification aisée, introduite par Bell dans les années 30 pour le téléphone, et répandue pour les ordinateurs sous forme de "mini-wrapping" (pattes de 0,6 mm au pas de 2,54 mm).

Table des figures

1.1 Balayage cavalier (Random-scan) 11
1.2 Tensions de déviation 12
1.3 Précision absolue 13
1.4 Génération des tensions Vx et Vy 13
1.5 Pipe-line d'animation 3D 15
1.6 Balayage TV (Raster-scan) 17
1.7 Courants de déviation 17
1.8 Bobines de Helmholtz 18
1.9 Signal Vidéo 19
1.10 Moniteur Vidéo 19
1.11 Synchronisation 20
1.12 Projection cinéma 23
1.13 Moirage 26
1.14 Echantillonnage spatial 26
1.15 Apparition de fréquence basse 27
1.16 Caractère alphanumérique 28
1.17 Palette 31
1.18 Normalisation du signal vidéo 32
1.19 Acquisition d'image 33
2.1 "Terminal" graphique 36
2.2 Simulation de terminal 37
2.3 Séquencement ligne du circuit EF9365 39

2.4 Gestion mémoire du circuit EF9365 40
2.5 Génération des adresses par le circuit EF9365 40
2.6 Génération de vecteurs 42
2.7 Tracé de Bresenham 42
2.8 Bresenham "câblé" 44
2.9 Clipping automatique 45
2.10 Générateur de caractères 45
2.11 Comparaison de performances 48
2.12 Vecteur et gestion de fenêtres 49
3.1 L'Alto, d'après [TML*82] 54
3.2 Le processeur de l'Alto, d'après [TML*82] 56
3.3 Le contrôleur graphique de l'Alto, d'après [TML*82] 58
3.4 Architecture du Dorado, d'après [Pie83] 60
3.5 Pipe-line du Dorado, d'après [Pie83] 61
3.6 Processeur du Dorado, d'après [Pie83] 62
3.7 Châssis du Dorado, d'après [Pie83] 63
3.9 Schéma du Macintosh, d'après [Wil84] 65
3.10 Pré-allocation des cycles mémoire du Macintosh 66
3.11 Le Blit, d'après [Pik84] 67
3.12 Schéma du Blit, d'après [Pik84] 68
3.13 Adressage mémoire sur SUN, d'après [BB80] 70
3.14 Contrôleur graphique du SUN, d'après [BB80] 70
3.15 Comparaison de performances, d'après [PGI84] 72
4.1 Chemin de données d'entrée mémoire 81
4.2 Chemin de données de sortie mémoire 81
4.3 Codage de la palette 83
4.4 Réalisation de la palette 84
4.5 Transcodage des pixels 85
4.6 Transcodage des pixels par une UAL 86
4.7 Commutation dynamique de table de transcodage 86
4.8 Remplissage de zone 87

4.9 Détection d'objet 87
4.10 Plan d'ensemble de la carte 88
4.11 Circuit CMG 90
4.12 Circuit H/C 91
4.13 Circuit MG 91
4.14 Circuit LME/P 92
4.15 Carte graphique VLSI 92
4.16 93
6.1 Bus synchrone multiplexé 110
6.2 Schéma de principe de la visu 111
6.3 Séquencement de la sortie du FIFO 112
6.4 Chemin de données complet 113
6.5 Séquenceur et générateur d'adresses 114
7.1 Circuit imprimé 122
7.2 Connexions dans un hypercube de dimension n 122
7.3 Carte massivement parallèle 124
7.4 Mémoires extérieures au circuit VLSI 125
7.5 Schéma du processeur élémentaire 126
7.6 Instruction à 2 adresses 128
7.7 Schéma du processeur d'adresses 130
7.8 Séquenceur du micro-contrôleur 131

Table des matières

Résumé 3

Mots clés 3

Chapitre 0: Introduction générale 7
Avertissement concernant l'utilisation de termes anglo-saxons 10

Chapitre 1: Introduction aux architectures de contrôleurs graphiques: historique, balayage cavalier et balayage télévision, contraintes temporelles 11

Historique: balayage cavalier (Random-scan) 11
Affichage en couleurs 15
Ecrans Tektronix à mémoire 16

Ecrans à balayage télévision (Raster-scan) 17

Différents problèmes liés au moniteur et au balayage 22
Scintillement de l'image lié à la fréquence trame 22
Correction "de gamma" 25

Anti-aliasing 25

Double-buffering 29

La palette et les convertisseurs digitaux / analogiques (DAC) 30

Chapitre 2: Les contrôleurs graphiques télévision orientés "vecteurs" 35

Présentation 35

Le premier co-processeur graphique télévision: EF9365 37
Raisons détaillées de certains choix 38

Structure du EF9365 39
Visualisation 39
Générateur de vecteurs 41
Flashback: l'algorithme de Bresenham 42
Réalisation de notre générateur de vecteurs 43
Clipping automatique 44
Générateur de caractères 45

Critique de ce circuit 46
Performances, utilisation en terminal 46
Utilisation en micro-ordinateur 47
Lien avec le BitBlT 49

Conclusion sur ce circuit 50

Les autres co-processeurs graphiques 51

Chapitre 3: Architectures de machines monochromes orientées BitBlT: Alto, Dorado, Macintosh, Blit, Sun, Apollo 53

Présentation 53

L'Alto 54
Structure du coupleur écran 57
Ecriture sur l'écran 58
Réalisation de la machine 59
Conclusions 59

Le Dorado 60

Le Macintosh 64
Détails du séquencement 66

Le Blit 66

SUN 1 69

SUN 2 71

Apollo 71

Circuits VLSI spécialisés 72

Performances comparées de ces machines pour l'exécution du BitBlT 72

Chapitre 4: Une réalisation personnelle déjà ancienne: carte 512 x 512, 16 couleurs, et sa critique actuelle 75

Présentation 75

Choix de l'architecture globale 77
Justification de ces choix 78

Organisation mémoire, parallélisme, double buffering, modularité du nombre de bits par pixel, chemin de données 79
Situation de la carte dans l'espace d'adressage du micro-processeur (problème du "bit-map") 82
Modularité réelle du nombre de bits par pixels 83

Palette et DACs 83

Processeur "pixel" 85

Plan d'ensemble 88

Découpage fonctionnel 89

Définition d'une famille de circuits VLSI possibles 90

Architecture de la maquette de simulation 93

Limitations et évolutions possibles 94

Critique actuelle de ce texte 95
Points positifs 96

Conclusion....97

Chapitre 5: Architectures spécialisées pour la synthèse d'image....99

La station IRIS....100

CUBI7....100

GETRIS....101

Machines pour le lancer de rayon....102

Machines cellulaires....103

Réflexion sur ces machines....103

Chapitre 6: Une visu sans mémoire d'image spécialisée, sur un ordinateur à bus modulaire....107

Présentation....107

Description du cadre de la réalisation....109

Description de la carte de visualisation....110

Séquenceur vidéo et générateur d'adresses....114

Réalisation physique de la carte et conclusions....115

Chapitre 7: Une petite machine massivement parallèle....117

Présentation....117

Concevoir une machine générale très puissante....118

Nécessité de paralléliser....119

Structure de notre machine POMP (petit ordinateur massivement parallèle)....123

Mise en boîtiers des processeurs élémentaires....124

Description du processeur élémentaire....127

Structure du micro-contrôleur....128

Etat actuel du projet et conclusions 132

Chapitre 8: Conclusion 133

Chapitre 9: Références bibliographiques..... 135

Glossaire..... 145

Table des figures..... 155

Table des matières..... 159