Conception d'un circuit intégré pour la visualisation graphique

Philippe Matherat

Philippe MATHERAT

CONCEPTION D'UN CIRCUIT INTEGRE

POUR LA

VISUALISATION GRAPHIQUE

Thèse de Troisième Cycle
19 Mai 1978

Résumé :

L'extension de l'utilisation des consoles graphiques est freinée par le prix élevé du matériel existant. Ce coût peut être abaissé par l'utilisation d'un téléviseur. Pour permettre le rafraîchissement d'un tel écran à balayage de trame, il est nécessaire de disposer d'une mémoire d'image où tous les points sont codés sur un bit en Noir et Blanc (ce qui correspond à 256 K bits pour 512 x 512 points), et davantage en couleur. Une telle capacité devient raisonnable vu l'augmentation de densité des mémoires intégrées, à condition que l'électronique de gestion de la mémoire d'image soit simple. L'intégration de celle-ci en un circuit LSI, comprenant en outre un générateur de vecteurs et de caractères câblé prévu pour un couplage par bus microprocesseur est l'objet de ce travail.

Le circuit permet une grande vitesse d'écriture (1,3μ s par point), et une grande versatilité, tant dans le format d'affichage (64 x 64 jusqu'à 512 x 512 points avec un nombre quelconque de niveaux de gris ou de couleurs) que dans le couplage microprocesseur (bus 8 bits bidirectionnel de données, bus d'adresse de 4 bits , signaux de lecture-écriture et requête d'interruption).

Dans le but de simplifier l'implantation topologique d'un tel circuit (équivalent à 2000 portes) très peu répétitif, la structure logique et géométrique d'un ensemble de fonctions de base ("briques") est proposée. La structure fonctionnelle de chacune des parties du circuit est ensuite étudiée, suivie de son implantation topologique, utilisant au mieux les briques précédemment définies.

Abstract :

Increased use of graphic display units is prevented by the high cost of existing devices. Lowering their costs suggests the use of TV sets. Display electronics for such a raster-scan unit will include a screen memory containing the state of all displayable points, that is up to 256 K bits for a 512 x 512 B&W display and more for colour applications. Technological advances in NMOS memories make such memory sizes cheaper and cheaper. The control part ensuring screen refresh, memory management, vector and character generation and computer coupling remains very complex. Design of a microprocessor-oriented Large-Scale-Integrated circuit including all these functions is the topic of this work.

The chip permits a decent writing speed of one point per 1,3μ s and a great versatibility, as well for screen memory organization as for microprocessor interface. Display resolution from 64 x 64 to 512 x 512 points are possible, with any number of colours or gray levels. Microprocessor interface is standard : an eight bits bidirectionnal data bus, a four bits address bus, read-write signals and an interrupt request.

In order to simplify the topological layout of such a circuit (equivalent complexity of 2000 gates) of a non-repetitive structure, logical and geometrical design of elementary building blocks is defined. Functional structure of each part of the circuit is then studied, followed by his topological layout using as much as possible the available building blocks.

Je remercie Monsieur le Professeur J. ARSAC d'avoir accepté de présider le jury de cette thèse.

Je remercie Messieurs F. ANCEAU et G. NOGUEZ pour leurs encouragements et leurs suggestions tout au long de ce travail.

Je remercie Monsieur C. GIRAULT pour ses critiques et indications lors de la rédaction.

Je suis très reconnaissant à Monsieur M. JOUMARD ainsi qu'à tous les membres de la SESCOSEM qui ont cru dès le départ à ce projet et grâce auxquels cette étude pourra donner lieu à une réalisation concrète.

Je suis très redevable à J.M. FRAILONG et J. GASTINEL pour les fructueuses discussions qui ont jalonné ce travail. Leurs remarques nombreuses m'ont guidé et c'est avec plaisir que je leur exprime ici mes remerciements.

Je ne saurais oublier de remercier Monsieur M. VALLINO, Directeur du Centre de Calcul de l'Ecole Normale Supérieure pour sa confiance, et les moyens mis à ma disposition, ni les usagers du centre, avec lesquels il m'a été agréable de travailler.

Je remercie enfin Monsieur LETELLIER du service des brevets de la THOMSON pour sa lecture différente du manuscrit et ses corrections.

CONCEPTION D'UN CIRCUIT INTEGRE POUR LA VISUALISATION GRAPHIQUE

0-INTRODUCTION

Les circuits intégrés réalisés en 1 985 atteindront une densité de 5 millions de transistors par boîtier pour des circuits d'organisation comparable aux microprocesseurs actuels. Une densité dix fois supérieure sera atteinte pour des circuits très répétitifs tels que des mémoires. Cette estimation, déduite de la loi de MOORE (1 964) -indiquant un doublement tous les ans depuis 1 95 9 et jamais démentie- sera certainement vérifiée [1]. En effet, non seulement les lois de la physique n'interdisent pas le fonctionnement de circuits de cette densité, mais les procédés de réalisation sont en cours de mise au point [2]. Ils consistent en le remplacement des rayons lumineux visibles par des électrons ou des rayons-X pour insoler la résine photosensible lors de la photolithographie. La finesse des motifs est alors augmentée dans le rapport des longueurs d'onde des rayonnements utilisés.

Or, une complexité de 5 millions de transistors correspond à un ordinateur IBM 370/168. Il est évidemment exclu que cet ordinateur soit un jour intégré tel quel pour constituer un seul composant, ne serait-ce qu'à cause du nombre trop élevé de connexions avec l'extérieur. La structure des futures machines sera très différente, fortement influencée par des "critères d'intégrabilité". Le parallèlisme sera largement favorisé car il augmente la répétitivité des circuits et facilite leur conception. Les algorithmes classiques ne permettront pas d'utiliser pleinement les possibilités de ces machines, et il faudra étudier des méthodes pour manipuler conceptuellement de telles structures. Ainsi, l'informatique sera probablement révolutionnée jusque dans ses aspects les plus théoriques.

En ce qui concerne la réalisation des ordinateurs, nous commençons à prendre conscience que leur coût n'est pas celui des éléments actifs (portes logiques), mais celui des connexions, tant entre les transistors d'une même "puce", que entre les puces [3]. Ce sont les connexions qui occupent le plus de surface, qui, par leur capacité électrique, diminuent la vitesse de fonctionnement, imposent la taille des transistors, et ainsi dictent la consommation électrique et la dissipation de chaleur. En outre, elles causent la quasi-totalité des pannes. Ceci justifie l'étude d'une nouvelle théorie des machines logiques, cherchant à minimiser non pas le nombre d'opérateurs de base nécessaires pour réaliser la fonction, mais la surface occupée par cette fonction. Il est en effet souvent intéressant de calculer plusieurs fois une variable logique en des "endroits" différents afin de supprimer les connexions nécessaires pour la transmettre. Une telle théorie tenant compte de considérations topologiques devrait manipuler comme un tout à la fois la fonction logique et l'implantation géométrique de cette fonction.

Avant d'étudier ces problèmes, nous avons beaucoup à faire pour nous mettre au courant des méthodes actuelles de réalisation des microcircuits, non pas les techniques chimiques et physiques de leur fabrication mais les méthodes de conception logique qui prennent place avant la réalisation des masques. Nous avons choisi à cette occasion de concevoir un composant qui, associé à un récepteur de télévision, permette de réaliser une console de visualisation graphique. Il est intéressant de constater que cette technique d'affichage (à balayage de trame) -qui oblige à mémoriser individuellement chaque point de l'écran pour les envoyer séquentiellement sous forme de signal vidéo- aurait parue tout à fait mal choisie il y a seulement 5 ans. Aujourd'hui, vu l'augmentation de densité des mé-

moires et la possibilité de réaliser toute l'électronique de gestion en un seul circuit intégré, il semble que ce soit le moyen qui permette, grâce à son faible coût, une réelle utilisation massive des consoles graphiques interactives.

En effet, un terminal interactif n'est efficace et ne fait gagner du temps à l'utilisateur que si celui-ci est présent devant l'écran pendant un temps assez long. Pourtant, aujourd'hui, dans les grands centres de calculs, il n'existe que quelques rares exemplaires de consoles graphiques, dont l'utilisation est facturée très cher.

Par ailleurs, ce besoin de consoles graphiques apparaît d'une autre façon: on commence à voir des miniordinateurs bon marché destinés à tous, comme les calculatrices de poche actuelles mais comportant un clavier et un écran de visualisation. Il est aisé de prédire à ces appareils une grande réussite commerciale. Ils pourront effectuer toutes sortes de traitements locaux, et pourront dans un avenir pas très lointain être connectés à de grands centres de "Time-Sharing" offrant leurs services par l'intermédiaire du téléphone (compagnies aériennes ou de chemins de fer pour les réservations, magasins de vente par correspondance, et surtout agences de presse, cours par correspondance, bibliothèques etc...) [4]. Ces miniordinateurs s'adresseront à des non-informaticiens, et devront être dotés de langages de haut niveau, utilisant largement l'affichage graphique [5]. Ceci ne sera possible qu'à la condition de disposer de consoles bon marché, mais possédant tout de même d'excellentes performances: définition suffisante, possibilités d'affichage en couleur et surtout une vitesse de tracé élevée. Cette dernière caractéristique permet l'affichage d'images en mouvement, et l'affichage rapide de figures fixes compliquées, ce qui est essentiel pour une

utilisation interactive efficace.

Le travail exposé ici présente la structure d'une unité de visualisation à balayage de trame, puis montre comment on peut en détacher une partie se prêtant bien à l'intégration, en tenant compte des contraintes de réalisation, mais aussi en recherchant la versatilité de l'objet qui devra ensuite être utilisé en tant que "boîte noire".

Le premier chapitre compare les différentes structures d'unités d'affichage et fixe les choix fondamentaux concernant la fonction du circuit intégré: circuit d'adressage d'une mémoire d'image graphique entièrement lue périodiquement lors de la visualisation, et accédée point par point en écriture par un générateur câblé de vecteurs et de caractères commandé de l'extérieur par un bus microprocesseur.

Les choix de détail dépendant de notre réalisation sont énumérés dans le second chapitre. Ils sont souvent basés sur une prévision des montages et des programmes utilisant le circuit et attendent une justification à postériori.

Avant de passer à une description précise du fonctionnement de chaque partie du circuit, le troisième chapitre expose la méthode d'implantation choisie qui dicte la structure logique des éléments, rejaillissant sur l'exposition des chapitres suivants. Ce chapitre se termine sur la proposition des schémas logiques des "briques" fonctionnelles d'une bibliothèque de base.

Les chapitres qui suivent (4,5,6, et 7) exposent en détails le fonctionnement des différentes parties du circuit et débouchent chacun sur deux schémas: l'un décrit le montage de simulation réalisé à partir de circuits existants, l'autre est la symbolisation à partir de portes élémentaires, permettant d' entreprendre le dessin des

masques du circuit intégré. Le chapitre 5 en particulier décrit la structure du générateur de vecteurs câblé qui adresse à chaque période d'horloge un nouveau point du vecteur à tracer. Nous montrons que l'approximation du vecteur obtenue est la même que celle de l'algorithme classique de génération programmée de vecteurs.

Une application de ce circuit à la simulation d'une console graphique très répandue est exposée dans le huitième chapitre.

1-CHOIX D'UNE STRUCTURE D'UNITE D'AFFICHAGE ADAPTEE A L'INTEGRATION

1.1-Constituants indispensables d'une unité d'affichage.

1.2-Les différents types d'écran.

1.3-Les différentes structures d'unité d'affichage.

a)Tube à mémoire et balayage cavalier.

b)Tube sans mémoire et balayage cavalier.

c)Tube sans mémoire et balayage de trame.

1.4-Avantages et inconvénients de chaque structure.

a)Résolution.

b)Possibilité d'affichage en couleurs.

c)Limitation du nombre de traits affichés.

d)Effacement sélectif.

e)Figures en mouvement.

f)Coût des composants.

1.5-Choix du balayage de trame.

1.6-Codage de la figure au niveau du terminal.

1.7-Quel type de mémoire utiliser pour le codage des points?

1.8-Choix des performances du système d'écriture.

1.9-Choix du dispositif de couplage avec l'extérieur.

1.10-Conclusion.

1.1-CONSTITUANTS INDISPENSABLES D'UNE UNITE D'AFFICHAGE

L'information graphique est rarement manipulée sous la forme d'une liste des états de tous les points composant une image. Tant dans le but de diminuer la quantité d'information à stocker ou à échanger que pour simplifier les algorithmes de traitement, on utilise un codage sous forme de listes de "vecteurs" et de "caractères". Un vecteur est défini par les coordonnées de son origine et de son extrémité ou par les longueurs de ses projections sur les axes de coordonnées. Un caractère est défini par son numéro d'ordre dans une liste (jeu de caractères). La manipulation d'une image se fait alors en termes d'ajouts ou de retraits de vecteurs ou de caractères.

Une unité d'affichage comporte donc toujours 3 éléments fondamentaux:

-l'écran, dont le rôle est d'émettre de la lumière,

-le générateur de vecteurs et de caractères, qui transforme la description codée en une description point à point,

-une mémoire, permettant de tenir à jour l'état de l'image affichée.

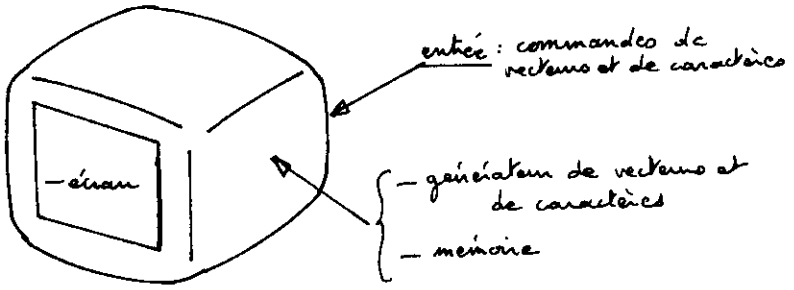


Fig. 1.1

1.2-LES DIFFERENTS TYPES D'ECRAN

La structure des 2 derniers éléments dépend fortement du premier. Aujourd'hui, toutes les unités d'affichage graphique disponibles sur le marché utilisent un tube à rayons cathodiques (CRT) [6]. De nombreux autres dispositifs sont à l'étude, basés sur les propriétés des cristaux liquides, des diodes électroluminescentes ou des plasmas, mais aucun n'a encore débouché sur une réalisation commerciale.

Ces CRTs, indépendemment de la possibilité éventuelle d'afficher des images en couleurs, peuvent se distinguer suivant 2 critères:

-certains sont "à mémoire". Dans ce cas, un point de l'écran atteint une seule fois par le faisceau d'électrons reste lumineux. L'image ne peut alors être effacée que sur toute sa surface en même temps, et cette opération est relativement longue (plusieurs dizièmes de secondes). Par contre, dans le cas d'un écran sans mémoire, un point n'est lumineux que pendant un court temps après le passage du faisceau d'électrons. Ce temps de "rémanence" est en général inférieur à 100µs -sauf dans le cas des radars- et inférieur au temps de parcours d'une ligne dans le cas de la télévision N&B. Le faisceau d'électrons doit alors constamment répéter son parcours pour "rafraîchir" l'écran, avec une période inférieure à la persistance rétinienne.

-le balayage du faisceau peut être "cavalier" ou "de trame". Dans le premier cas, le faisceau ne parcourt que les points à allumer et à intensité constante (sauf pour sauter d'une partie de la figure à une autre non connexe). Dans le cas du balayage de trame, la totalité de la surface de l'écran est constamment balayée par le faisceau qui suit toujours le même chemin quelquesoit la figure affichée. Il faut alors moduler l'intensité du faisceau en fonction

de l'appartenance du point atteint à la figure affichée. Le parcours s'effectue généralement de gauche à droite à une fréquence voisine de 20kHz et de haut en bas à une fréquence voisine de 50Hz.

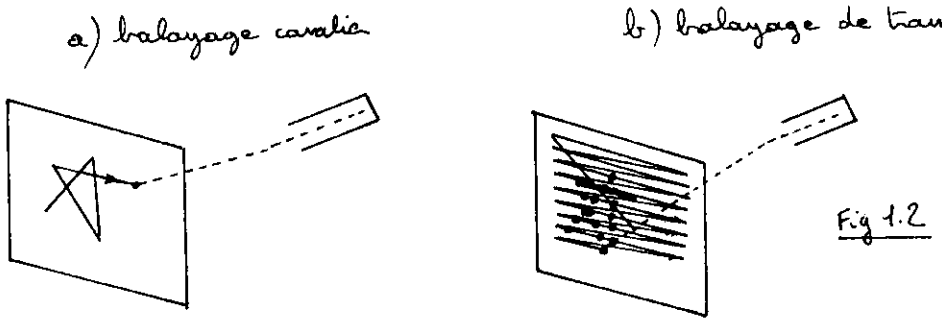


Fig 1.2

1.3-LES DIFFERENTES STRUCTURES D'UNITE D'AFFICHAGE

En fonction de ces 2 critères de distinction des CRTs, on peut classer les unités d'affichage en 3 groupes dans lesquels les constituants fondamentaux sont organisés différemment:

a)<u>Tube à mémoire et balayage cavalier.</u> C'est la structure des consoles TEKTRONIX de la série 4010 [22].

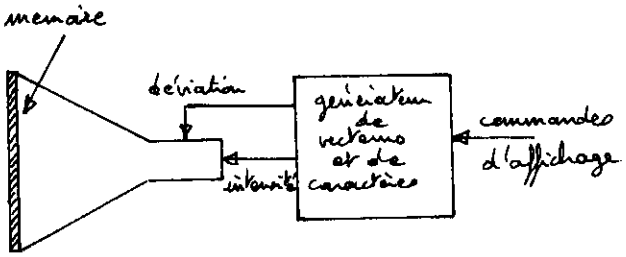


Fig 1.3

-L'écran mémorise l'image point à point.

-Le générateur de vecteurs et de caractères commande la déviation du faisceau.

b)<u>Tube sans mémoire et balayage cavalier.</u> C'est en particulier la structure des unités IBM 2250, et du mode "non marquant" des TEKTRONIX du haut de la gamme 4010 (dans ce mode, et sur ces modèles, l'énergie fournie par le faisceau d'électrons est choisie faible pour éviter d'inscrire l'image dans la mémoire du tube).

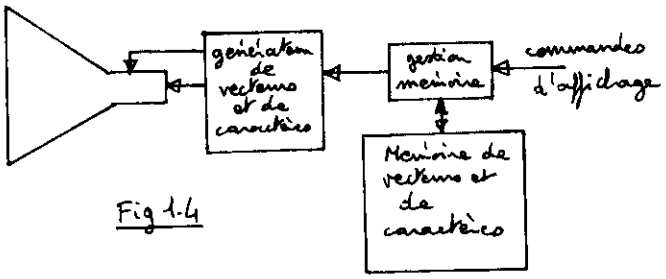


Fig 1.4

-L'image est codée en mémoire sous forme d'une liste de vecteurs et de caractères.
-Le générateur de vecteurs et de caractères commande la déviation du faisceau.

c)Tube sans mémoire et balayage de trame. C'est le cas des nouvelles unités HEWLETT-PACKARD 2648A [10] et TEKTRONIX de la série 4020.

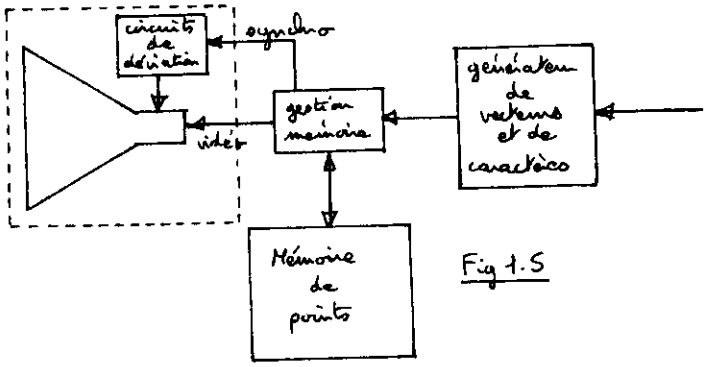


Fig 1.5

-L'état de chacun des points est mémorisé.
-Le générateur de vecteurs et de caractères est constitué de circuits digitaux modifiant la mémoire de points.

Il faut adjoindre dans ce cas des circuits de déviation (synchronisés avec la lecture de la mémoire de points) puisque le trajet du spot est indépendant de la figure affichée. Un récepteur de télévision réalise la fonction encadrée en pointillés sur la figure 1.5.

Un écran à mémoire associé à un balayage de trame présente peu d'intérêt, puisque l'affichage d'une seule trame nécessite déjà la mémoire de points.

1.4-AVANTAGES ET INCONVENIENTS DE CHAQUE STRUCTURE

a)Résolution. La précision des tubes à balayage cavalier est toujours supérieure à 1000x1000 points. Celle des tubes à balayage de trame est limitée verticalement par le nombre de lignes, horizontalement par la bande passante des circuits électroniques, et peut difficilement atteindre cette valeur. Elle est aussi limitée par la dimension et le coût de la mémoire de points, mais cette limite disparaît aujourd'hui puisqu'elle rejoint la définition de 1000x1000points.

b)Possibilité d'affichage en couleur. Les tubes à mémoire en couleurs n'existent pas.

c)Limitation du nombre de traits affichés. Le cas des tubes à balayage cavalier sans mémoire est particulier: la fréquence de rafraîchissement est liée à la longueur totale des traits affichés (ou au nombre de traits suivant le cas). Au delà d'une certaine complexité, l'image scintille.

d)Effacement sélectif. Les écrans à mémoire ne permettent pas d'effacer séparément un trait de la figure.

e)Figures en mouvement. Les écrans à mémoire ne le permettent pas. Les écrans à balayage de trame sont limités par la vitesse de mise à jour de la mémoire de points.

f)Coût des composants. Le coût comprend celui du tube, plus élevé dans le cas des tubes à mémoire, et le coût de l'électronique de commande, plus élevé dans le cas du balayage de trame.

1.5-CHOIX DU BALAYAGE DE TRAME

L'argument concernant le coût est celui qui fait le plus évoluer la structure des unités d'affichage. En effet, les tubes à

balayage de trame ont pour seul défaut intrinsèque une résolution légèrement plus faible. Or, ils ont été longtemps rejetés du fait de la complexité plus grande de l'électronique de commande. Aujourd'hui, la situation change dans la mesure où ces circuits sont intégrables, et où le coût d'un montage électronique dépend davantage du nombre de ses composants que de la nature de ceux-ci. En outre, la structure des unités à balayage de trame a l'énorme avantage de coïncider en partie avec la fonction réalisée par les téléviseurs, ce qui permet un coût encore plus faible vu le marché de ces appareils, sans compter qu'une grande partie des applications concerne l'informatique à domicile, où les téléviseurs existent déjà. D'autre part, l'importance de ce marché garantit que tous les futurs procédés physiques d'affichage seront adaptés aux téléviseurs et donc aux consoles qui les utilisent.

1.6-CODAGE DE LA FIGURE AU NIVEAU DU TERMINAL

Nous avons dit que dans le cas du balayage de trame, la mémoire est une mémoire de points. Ceci est dicté par le problème suivant: le signal vidéo représente à tout moment la fonction logique: {le point atteint par le faisceau d'électrons appartiend à la figure affichée}. Pour une définition horizontale de 500 points, cette fonction doit être évaluée tous les 100ns. Le codage de la figure au niveau du terminal doit permettre d'atteindre cette vitesse.

Si la figure est codée sous forme d'une liste de vecteurs, l'évaluation de cette fonction au point de coordonnées (x,y) se fait par l'exécution de la séquence suivante: (impossible en 100ns)

début vidéo:=faux ; pour i depuis 1 jusqu'à n tantque ⌐vidéo
faire vidéo:=$(x\in[x_{oi}, x_{oi}+\Delta x_i])$ et $(y\in[y_{oi}, y_{oi}+\Delta y_i])$ et $(\frac{x-x_{oi}}{\Delta x_i}=\frac{y-y_{oi}}{\Delta y_i})$
fait fin

(où la liste comprend n vecteurs, définis par leurs coordonnées d'origine (x_{oi},y_{oi}) et leurs projections sur les axes $(\Delta x_i ,\Delta y_i)$).

Pour certains types de figures, il est possible de trouver un codage plus dense. En effet, si toutes les figures représentées sont constituées de la répétition sans rotations ni homothéties d'un petit nombre de petites sous-figures, alors il est possible de coder l'image sous forme de 2 listes: une liste donnant la définition point par point des sous-figures, une autre liste donnant les coordonnées des occurences de ces sous-figures [7]. Un câblage spécifique, adapté aux dimensions de chacune des listes, peut générer le signal vidéo assez rapidement.

Mais cette méthode n'est pas utilisable pour une unité d'affichage à usage général, car elle restreind l'ensemble des figures affichables à une classe particulière.

1.7-QUEL TYPE DE MEMOIRE UTILISER POUR LE CODAGE DES POINTS?

Les mémoires les plus denses disponibles actuellement sont des 64 Kbits CCD (charge coupled devices). Ces circuits sont organisés en registres à décalage et sont particulièrement adaptés à une sérialisation des données. Ceci est un avantage pour la génération du signal vidéo. Mais l'écriture de nouveaux éléments dans cette mémoire devient alors très problématique. Deux solutions se présentent: On peut attendre qu'une adresse soit utilisée en lecture pour modifier le contenu. Ceci limite alors la vitesse d'écriture des vecteurs à 1 par trame (50/s). Une autre solution consiste à utiliser un "buffer" intermédiaire dans lequel écrit le générateur de vecteurs et de caractères, et qui est transféré à la mémoire lors de la visualisation.

Devant les inconvénients de ces solutions, nous avons préféré

coder les points de l'image dans des mémoires à accès aléatoire (RAM). La densité actuelle est 16 Kbits/boîtier, ce qui correspond à 16 boîtiers pour un affichage 512x512 N&B. La lecture séquentielle périodique de visualisation permet de ne pas se poser le problème du rafraîchissement de ces mémoires dynamiques.

1.8-CHOIX DES PERFORMANCES DU SYSTEME D'ECRITURE

Toutes les réalisations actuelles utilisant une telle mémoire d'image et destinées à des affichages de faible coût n'offrent comme possibilités d'écriture que de modifier le point adressé par 2 registres X et Y, seuls registres accessibles par une unité de traitement (c'est le cas notamment des VRAMs de MATROX [9] et des circuits alphanumériques utilisés pour l'affichage graphique). Il faut alors tracer tous les vecteurs et les caractères à l'aide de générateurs programmés. Outre la place mémoire occupée par ces programmes, le temps d'écriture est trop grand (au moins 50µs/point) incomparable avec celui des unités d'affichage de qualité (≃50µs par vecteur). Il est indispensable pour une utilisation informatique de disposer d'un générateur de vecteurs et de caractères câblé.

Nous n'avons pas implémenté de générateur d'arcs de cercles. Cela nécessiterait une transformation des paramètres avec laquelle la structure choisie est peu compatible. Il serait intéressant d'envisager cette extension en prévision de l'augmentation des densités d'intégration. A notre avis, le paramétrage intéressant serait: la tangente initiale (1) (donnée par les paramètres du vecteur précédent), le rayon de courbure (2),et la tangente finale (3).

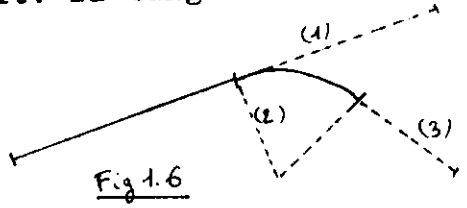


Fig 1.6

1.9-CHOIX DU DISPOSITIF DE COUPLAGE AVEC L'EXTERIEUR

Nous avons choisi un mode de couplage par bus microprocesseur. Aujourd'hui, un microprocesseur est d'utilisation aussi simple que tout autre composant. Son ajout indispensable à l'unité d'affichage n'enlève pas de possibilités à celle-ci, mais permet l'interfaçage simple avec toutes sortes de coupleurs. Ce microprocesseur pourra être un dispositif de transcodage pour émuler le code des autres unités existantes et permettra ainsi d'utiliser leurs logiciels (voir chapitre 8). Il pourra être le principal organe de traitement dans le cas de petits systèmes, ou un organe permettant de décharger un gros ordinateur sur lequel l'unité est connectée. En particulier, si cette connexion est une ligne à basse vitesse (réseau téléphonique commuté par exemple), le microprocesseur peut être un instrument de compression-décompression de l'information, qui associé à un programme semblable sur le gros ordinateur, permet d'utiliser au mieux les possibilités de la ligne.

1.10-CONCLUSION

Tous ces choix nous conduisent à une structure qui peut se schématiser ainsi: (nous avons ajouté un circuit de gestion d'un "light pen")

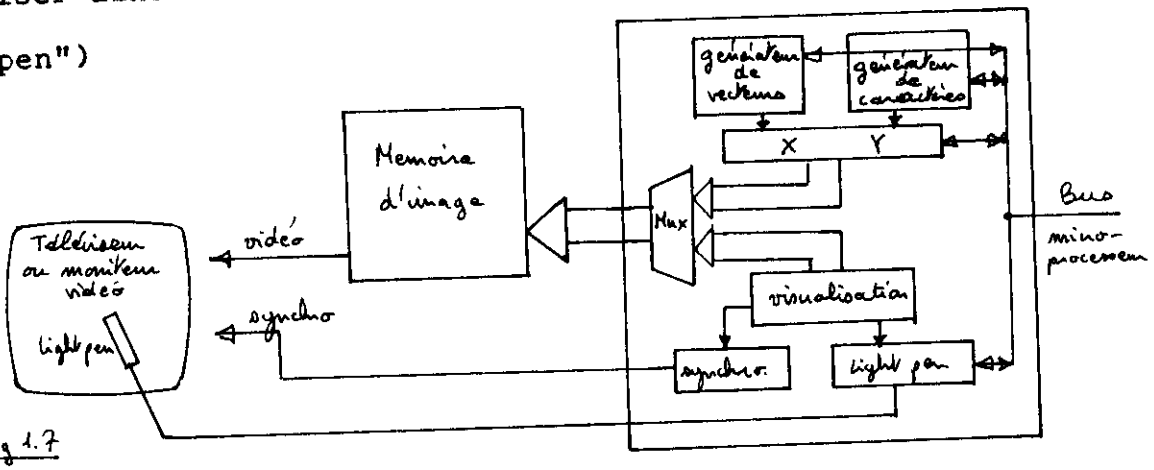


Fig 1.7

Nous proposons d'intégrer la partie encadrée. C'est un ensemble qui est nécessaire tel quel à toutes les unités d'affichage à balayage de trame, alors que la mémoire d'image par contre dépend des particularités de chaque réalisation: résolution, couleurs, niveaux de gris, superposition d'images, etc...

Cette structure de circuit, tout en offrant un maximum de possibilités, en particulier une grande vitesse d'inscription, ne pénalise pas les utilisations simples où le générateur de vecteurs et de caractères est inutile. Il mérite donc d'être considéré comme un composant à part entière, se définissant indépendemment d'une utilisation particulière.

2-DESCRIPTION FONCTIONNELLE DU CIRCUIT

2.1-Rappel: codage de l'information pour un téléviseur.

2.2-Rappel: utilisation des mémoires dynamiques 16 broches 4Kx1bits et 16Kx1bits.

2.3-Organisation de la mémoire d'image.

2.4-Sélection des boîtiers mémoires.

2.5-Signaux de contrôle mémoire.

2.6-Exemples de montages (Noir et Blanc).

a)64x64 points.

b)128x128 points.

c)256x256 points.

d)512x512 points.

e)Autres formats.

2.7-Signal de synchronisation.

2.8-Récapitulation pour la partie visualisation.

2.9-Système d'écriture.

2.10-Registres X et Y.

2.11-Générateur de vecteurs.

2.12-Générateur de caractères.

2.13-Light pen et Réticule.

2.14-Mécanisme d'effacement de l'image entière.

2.15-Mécanisme d'interruption.

2.16-Lecture d'un point de la mémoire d'image.

2.17-Liaison au bus microprocesseur.

2.18-Récapitulation des codes.

Le but de ce chapitre est d'énumérer les particularités du circuit réalisé avant de passer à l'étude de son fonctionnement et de sa conception logique proprement dite. Certaines caractéristiques du circuit sont imposées: les fréquences de balayage ligne et trame et la forme des signaux de synchronisation sont imposés par l'utilisation d'un téléviseur standard; l'adressage de la mémoire d'image est imposé par l'utilisation des mémoires les plus intégrées disponibles actuellement. Les autres caractéristiques ont fait l'objet de choix guidés par des essais de montages et de programmes d'application.

2.1-RAPPEL: CODAGE DE L'INFORMATION POUR UN TELEVISEUR [11]

Le faisceau d'électrons balaye constamment la totalité de la surface de l'écran suivant le trajet indiqué en figure 2.1.

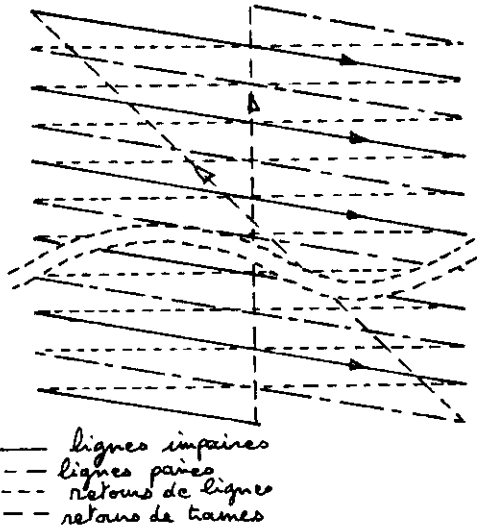


Fig 2.1

L'écran est parcouru en entier 25 fois par sec. Chacun de ces parcours ("image") comporte 625 lignes, tracées en 2 passes ("trames") de 312,5 lignes, une trame pour les lignes paires, une pour les lignes impaires. On dit qu'il y a "entrelacement" des trames.

A chaque instant, un signal de "luminance" indique l'intensité du faisceau d'électrons correspondant à sa position. La déviation du faisceau est assurée par 2 bases de temps incluses dans le récepteur de télévision auxquelles il faut fournir des signaux de

synchronisation pour ajuster précisément leurs périodes et leurs phases.

2 signaux sont donc nécessaires pour commander un récepteur TV N&B: -la luminance (signal "vidéo"),

-la synchronisation (signal "synchro").

(Ces 2 signaux sont additionnés pour constituer le signal "vidéo composite" dans le cas de l'émission sur une onde porteuse UHF).

Les "tops" nécessaires à la synchronisation sont les suivants:

-un retour ligne est commandé par un top ligne de durée 4,5µs toutes les 64µs ($=\frac{1}{25 \times 625}$s) Fig 2.2.a,

-un retour trame est commandé par un top trame d'une durée de 2,5 lignes toutes les 20ms Fig 2.2.b.

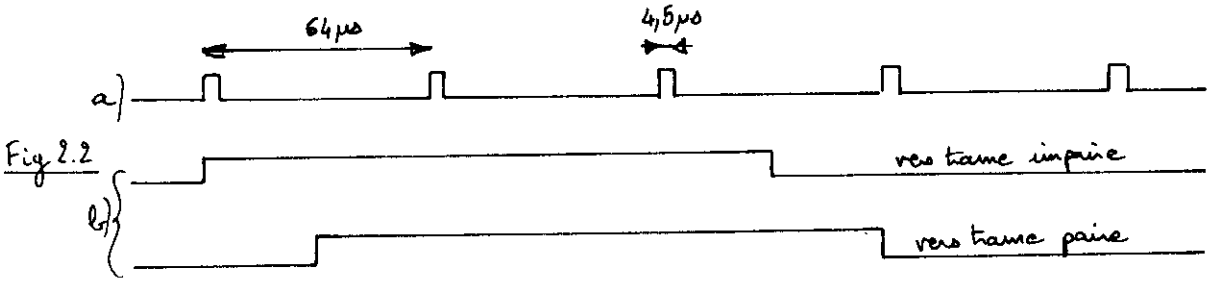


Fig 2.2

Pour assurer l'entrelacement, un top trame sur deux doit commencer en même temps qu'un top ligne, un sur deux doit commencer en milieu de ligne.

Pour transmettre un signal unique de synchro, on utilise le OU logique de ces 2 signaux. Le téléviseur les "trie" de la façon suivante:

-un intégrateur suivi d'un circuit à seuil permet de reconnaître le top trame (Fig 2.3). La sortie S commande directement le retour trame.

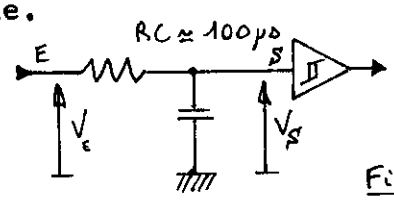
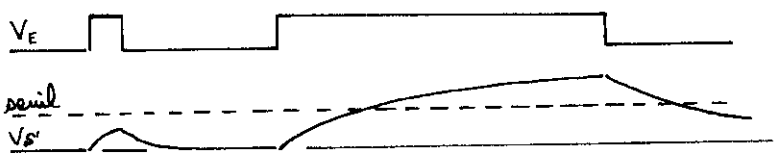


Fig 2.3



-un différentiateur permet de reconnaître le top ligne (Fig 2.4). La sortie S ne commande pas directement le retour ligne, mais asservie l'oscillateur ligne par l'intermédiaire d'un comparateur de phase.

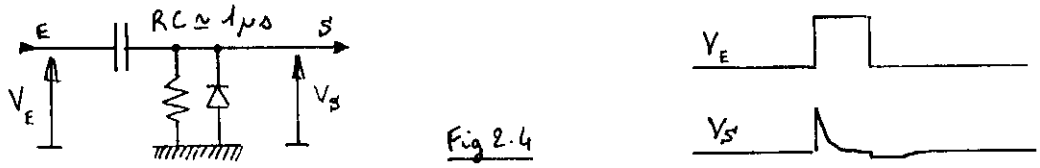


Fig 2.4

Le signal de la Fig.2.2 n'est pas satisfaisant car:

-l'information de synchro ligne disparaît lors du top trame,

-l'intégrateur n'est pas dans le même état avant un top trame si celui-ci arrive 64µs ou 32µs après un top ligne.

Pour ces raisons, le signal de synchronisation du standard français est le suivant:

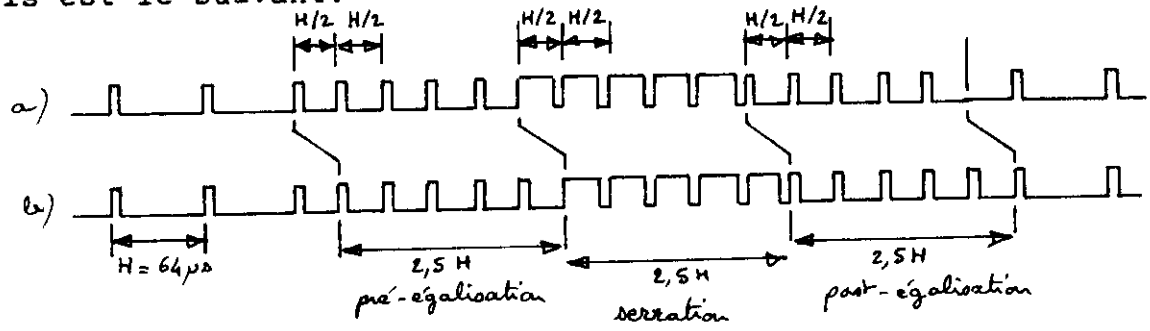


Fig 2.5

Ceci permet de garder l'asservissement de l'oscillateur ligne lors d'un top trame, et de présenter à l'intégrateur un top trame toujours environné de la même façon.

Si l'on veut décrire une image dont la définition verticale est inférieure à $\frac{625}{2}$ lignes, il est recommandé de réaliser 50 trames identiques par secondes, c'est à dire de cadrer tous les tops trames de la même façon par rapport aux tops ligne. L'image, non entrelacée, paraît plus stable sur l'écran. Le signal de synchro peut être simplifié, comme par exemple ([8]) en Fig.2.6, mais il peût aussi être

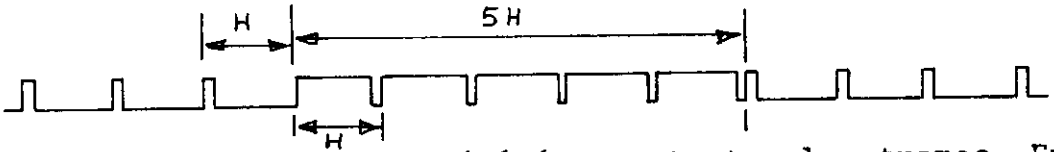


Fig 2.6

l'un des signaux de la Fig.2.5, répété pour toutes les trames. En particulier, nous avons utilisé celui de la Fig.2.5.b.

En couleur, le signal de synchronisation est identique. Le signal vidéo porte une information de luminance et une de chrominance. Le codage utilisé en télévision ne permet pas une définition supérieure à 160x160 points pour l'information de couleur. On doit alors commander directement les 3 canons à électrons Rouge, Vert et Bleu (entrée R,V,B+Synchro) si l'on désire une définition de 500x500 points pour la couleur.

2.2-RAPPEL: UTILISATION DES MEMOIRES DYNAMIQUES 16 BROCHES 4Kx1BITS ET 16Kx1BITS [12], [13]

Ce sont des mémoires dynamiques à adressage multiplexé.

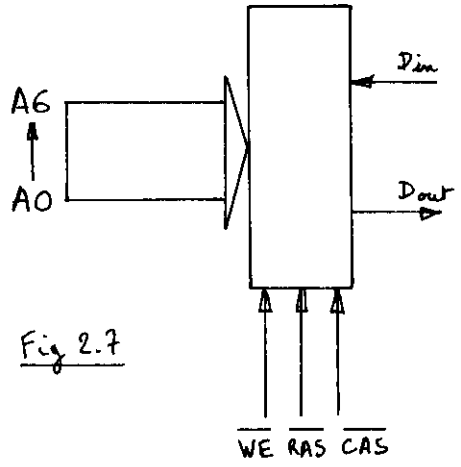


Fig 2.7

La mémoire 16K est internement organisée suivant une matrice de 128 rangées et 128 colonnes.

Le front avant de $\overline{RAS}$ (Row address select) échantillonne la première partie de l'adresse (ou partie basse) -voir Fig.2.8.

Le front avant de $\overline{CAS}$ (Column address select) échantillonne la deuxième partie (haute) de l'adresse.

Dans le cas de la 4K, A6 sert de sélection de boîtier ($\overline{CS}$, actif bas):-il est ignoré sur $\overline{RAS}$,

-il est échantillonné sur $\overline{CAS}$.

Il faut toujours avoir à l'esprit que la mémoire contient autant d'amplificateurs de rafraîchissement que de colonnes, si bien que lors d'un accès, toute une rangée est rafraîchie. Dans le cas d'un accès séquentiel à tous les bits, le bon rafraîchissement s'opère

donc en faisant varier plus rapidement les adresses de rangées que celles de colonnes.

Par la suite (chapitre 4), nous nommerons les bits d'adresse:

R0,R1,R2,...,R6,C0,C1,...,C6.

Pour généraliser (en oubliant les petites variantes entre les mémoires des différentes firmes), les chronogrammes d'accès sont les suivants:

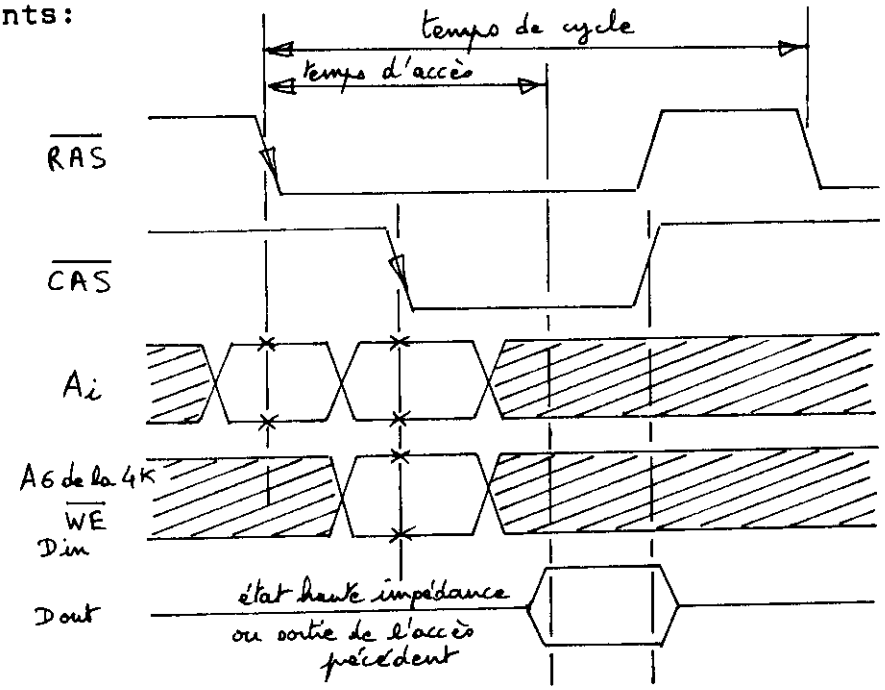


Fig 2.8

-$\overline{WE}$ ($\overline{Write\ Enable}$) indique s'il s'agit d'un cycle de lecture ou d'écriture.

-La donnée d'entrée (Din) est échantillonnée sur $\overline{CAS}$.

-Dans tous les cas, si $\overline{RAS}$ ou $\overline{CAS}$ manque, le boîtier n'est pas concerné par l'accès mémoire. Nous utiliserons cette propriété pour sélecter les boîtiers par l'intermédiaire du $\overline{RAS}$, le $\overline{CAS}$ étant constamment généré pour tous les boîtiers.

2.3-ORGANISATION DE LA MEMOIRE D'IMAGE

Le signal vidéo décrit successivement l'état de chacun des points d'une même ligne, et nous avons vu qu'à une définition

horizontale de 500 points correspond une période de 100ns par point environ. Le temps de cycle des mémoires actuelles étant de 350ns, il est impératif de lire plusieurs points en même temps, différants uniquement par la partie faible de leur adresse horizontale, et de les sérialiser grâce à un registre à décalage pour former le signal vidéo. La mémoire est donc organisée en mots:

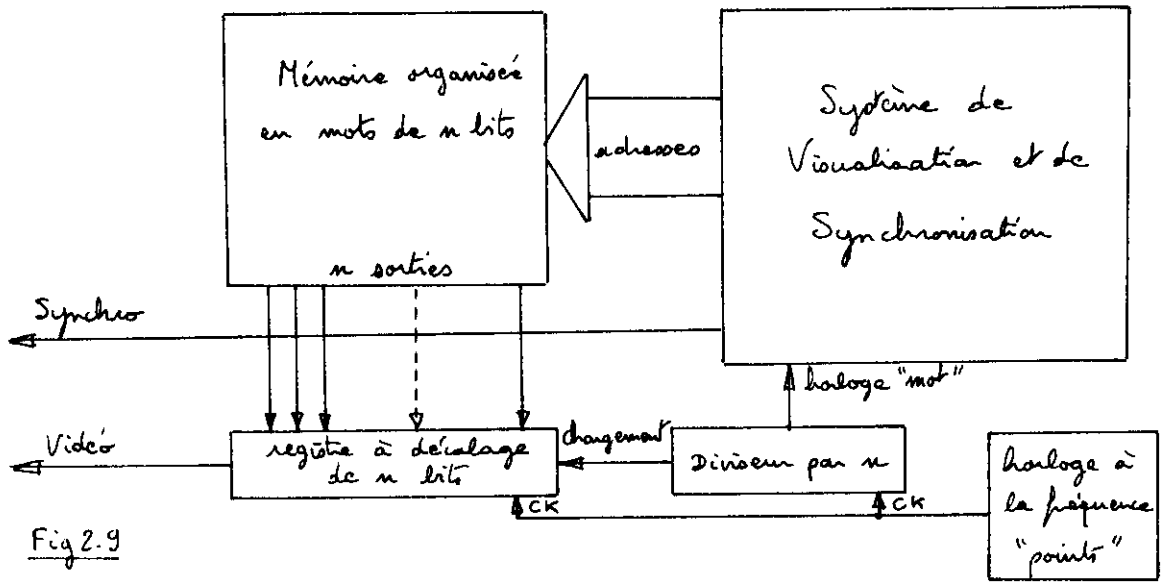


Fig 2.9

Il est nécessaire de disposer d'une horloge tournant à la fréquence de succession des points, pour décaler le registre. La fréquence de modification des adresses mémoire et du signal de chargement du registre à décalage doit, elle, être n fois plus faible (si n est le nombre de bits du registre à décalage).

Changer la définition horizontale de l'image peut alors se faire de 2 façons différentes:

-on peut conserver le même registre à décalage et le même diviseur, et changer la fréquence de l'horloge "points". Il faut alors changer le fonctionnement du système de visualisation car le nombre de périodes par ligne TV sera différent.

on peut aussi changer la longueur du registre à décalage et le facteur de division du compteur proportionnellement à la définition horizontale, ce qui laisse constante la fréquence de fonctionnement

du système de visualisation et le nombre de périodes par lignes.

Nous avons choisi la deuxième solution, en excluant du circuit intégré la mémoire, le registre à décalage, le diviseur, et l'horloge points, ce qui rend le circuit très peu dépendant du format de visualisation, tout en permettant des applications comprenant peu de boîtiers (Fig.2.15 à 2.18).

La fréquence de fonctionnement du circuit, qui est aussi celle des accès mémoire, est choisie la plus élevée possible compte tenu de la technologie de réalisation (1,75MHz, ce qui correspond à 550ns de cycle). Comme cette fréquence est aussi celle d'accès à un point en écriture, ceci garantit une vitesse de tracé de nouvelles figures maximale (vecteurs,caractères). Le nombre de périodes de visualisation est 64 par lignes, ce qui en laisse 48 pour l'écriture. La fréquence d'écriture est donc de $1{,}75 \times \frac{48}{112}$MHz (environ 1,3µs/point), et peut atteindre 1,75MHz si on "force en écriture" (voir broche FECR §2.8).

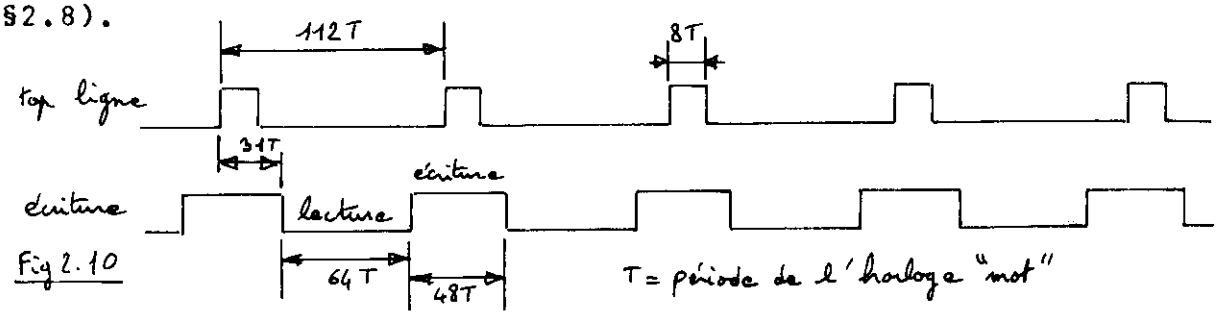


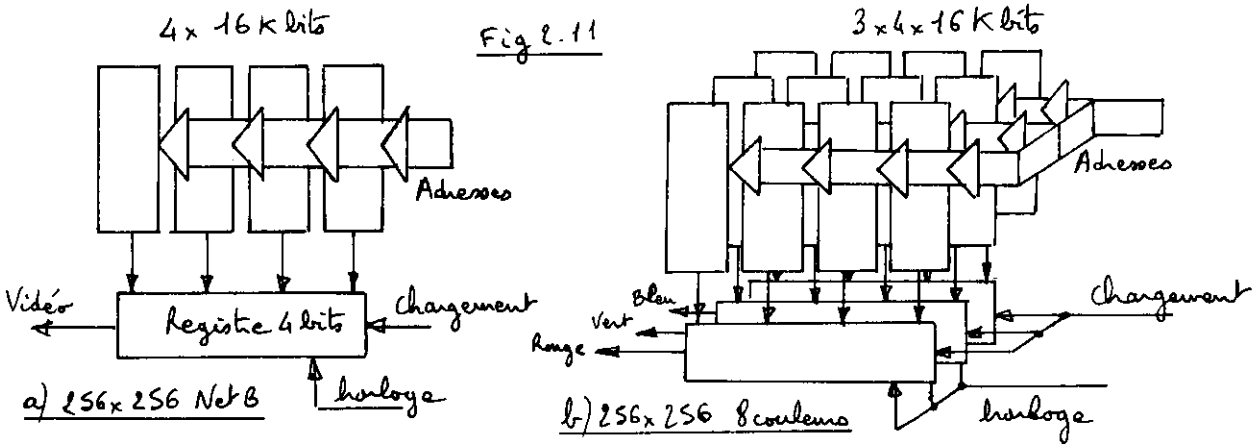
Fig 2.10

L'organisation de la mémoire en fonction du format est la suivante:

Définition:	n:	boîtiers nécessaires:	organisation:
512x512	8	16x16Kbits	32 Kmots de 8 bits
256x256	4	4x16Kbits	16 Kmots de 4 bits
128x128	2	4x 4Kbits 2x 8Kbits	8 Kmots de 2 bits
64x64	1	1x 4Kbits	4 Kmots de 1 bit.

Pour augmenter le nombre de bits nécessaires pour décrire un point (couleurs, dégradés, superpositions), l'ensemble mémoire-registre doit être multiplié par ce nombre de bits. Mais la longueur

des mots mémoire et leur nombre restent constants. Il faut considérer que la mémoire croît dans une troisième dimension:



Pour une application avec niveaux de gris, les 3 sorties série commandent un convertisseur digital-analogique 3 bits. Pour une superposition, on utilise un OU logique.

2.4-SELECTION DES BOITIERS MEMOIRE

Ce mode d'adressage de la mémoire ne suffit pas. En effet, en écriture, le générateur interne de vecteurs et de caractères accède les points un par un. Il faut donc utiliser la partie basse de l'adresse horizontale d'écriture pour choisir le bit concerné dans le mot mémoire. Comment notifier cette sélection de boîtier à l'extérieur du circuit intégré, alors que le nombre de boîtiers mémoire change suivant le format? (Fig.2.15 à 2.18)

Considéront pour simplifier, uniquement les formats 512x512 (16 boîtiers) et 256x256 (4 boîtiers). Il est exclu de mobiliser 16 broches du circuit intégré pour cette sélection. Nous avons choisi de n'utiliser que 4 broches, l'adresse faible horizontale (sur 4 bits) sortant directement pour le format 512x512, et décodée pour le format 256x256 (en forme de $\overline{RAS}$ pour les connecter directement aux boîtiers). La fonction de ces 4 broches (appellées ISL0,1,2,3) dépend d'une

entrée (FMAT) spécifiant en outre au générateur de synchronisation s'il doit fonctionner de façon entrelacée ou non.

En 512x512, il faut donc adjoindre un décodeur extérieur.

Pour la lecture, il est nécessaire de sélecter ensemble tous les bits d'un mot. En 256x256, les signaux de sélection ISLi s'en chargent. En 512x512, un signal supplémentaire ($\overline{IALL}$) indique les phases de lecture et doit forcer la moitié des sorties du décodeur au niveau bas (Fig.2.18).

Dans les 2 formats, la portion visualisée de l'espace logique adressable est différente, mais la précision de visualisation est la même que la précision logique utilisée pour décrire les dessins (voir §2.10).

Considérons maintenant le cas des formats inférieurs à 256x256. Comme la synchronisation est identique au cas 256x256, la patte d' entrée FMAT doît être au même niveau logique. Les sorties ISLi sont regroupées 2 par 2 en 128x128 et les 4 ensemble en 64x64 (dans ce cas d'ailleurs les signaux ISLi sont inutiles, voir Fig.2.15). De la même façon, l'adressage vertical se fait en ignorant 1 bit en 128x128 puis deux en 64x64 (Fig.2.16). La précision de visualisation n'est plus la précision logique. Par exemple, un point de l'image 128x128 est le OU logique de 4 points de l'image 256x256.

Pour résumer, on peut dire que dans le format 256x256 (et inférieurs), l'adresse de point est constituée de 16 bits dont 2 sont décodés sur ISLi , et les 14 autres envoyés en deux coups sur IADi. Dans le format 512x512, l'adresse comprend 18 bits dont 4 sont envoyés sur ISLi, et les 14 autres sur IADi.

2.5-SIGNAUX DE CONTROLE MEMOIRE

Les adresses sont envoyées en 2 coups, conformément aux

spécifications des mémoires. Elles sont donc issues d'un multiplexeur à 8 entées, commandé par 3 signaux:

-FMAT,

-lecture/écriture (=$\overline{IALL}$),

-partie haute/partie basse.

Le signal FMAT est nécessaire au niveau de ce multiplexeur, d'une part pour changer la partie de l'espace adressable mémorisée, et surtout pour adapter au rafraîchissement des mémoires (voir schémas d'applications, Fig.2.15 à 2.18).

Le signal de commutation entre partie haute et partie basse est le signal d'horloge du boîtier (broche CKIN). La modification du rapport cyclique de celui-ci entraîne la modification de l'instant de commutation entre ces adresses. Ceci permet d'utiliser toutes les mémoires ayant un temps de cycle inférieur à la période de fonctionnement du circuit intégré (550ns).

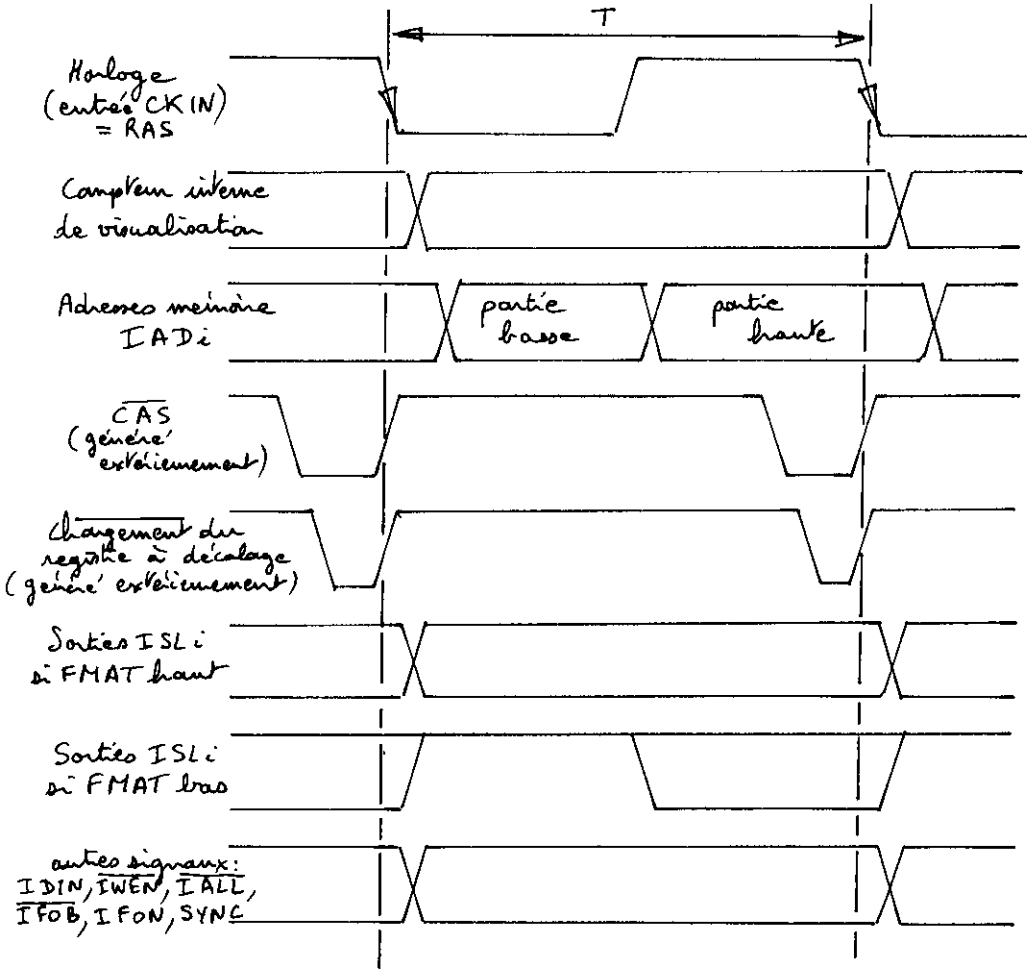


Fig 2.12

Le signal $\overline{IALL}$ est bas lors des phases de lecture ou de rafraîchissement. Il est identique à toutes les lignes et peut être utilisé comme signal de cadrage horizontal. Le circuit intégré commande en outre le signal d'écriture $\overline{WE}$ des mémoires (broche $\overline{IWEN}$) -ce signal n'est bas que pour les périodes d'écriture effective dans la mémoire-, ainsi que la donnée d'entrée des mémoires (broche IDIN) qui change en fonction d'un bit du mot de contrôle (voir §2.10) et peut aussi être forcée lors d'un cycle d'effacement. Ce signal sur 1 bit n'est pas particulier à la visualisation en Noir et Blanc. En effet, dans les visualisations en couleurs par exemple, un des états des points est particulier: l'état "éteint". La sortie IDIN doit être considérée comme une sortie de forçage de la donnée mémoire à l'état éteint (actif haut).

Il faut ajouter à ces signaux deux sorties pour forcer à volonté le signal vidéo dans un état déterminé. Ce sont le forçage à noir (IFON) et le forçage à blanc ($\overline{IFOB}$). IFON sert à nettoyer le signal vidéo sur les bords de l'écran ("blanking"), quand les sorties des mémoires peuvent présenter des états indésirables (rafraîchissement, écriture). Ceci permet d'ailleurs d'adapter toutes sortes de mémoires, même à entrées-sorties multiplexées. $\overline{IFOB}$ sert au début de chaque trame dans la partie non visible de l'écran, pour forcer le signal vidéo à blanc pendant une ligne, conformément aux normes. Il permet en outre de forcer l'écran à blanc pour simplifier l' utilisation du "Light pen" (§2.13).

Note: occupation de l'écran: la zone de rafraîchissement (qui ne se distingue de celle de visualisation que par le niveau haut du signal IFON) est indispensable. Elle ne peut pas par exemple être intégralement transformée en écriture. En effet, le temps correspondant aux marges haute et basse est de l'ordre de $64x(312-256)\mu s \simeq 3,5ms$ ce qui est trop long sans rafraîchissement (max: 2ms).

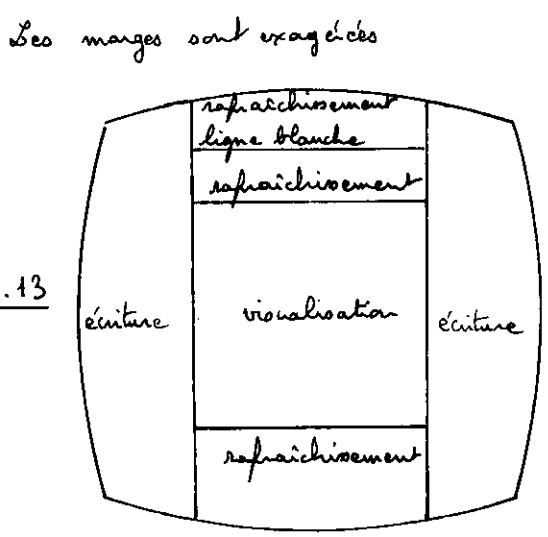


Fig 2.13

La ligne blanche en haut a lieu pendant le temps de forçage à noir. Or, il est plus facile (voir Fig.2.15 à 2.18) de réaliser un forçage à noir prioritaire sur le forçage à blanc que l'inverse. Pour cette raison, IFON devient bas lors de cette ligne. Dans ces conditions, les forçages à blanc et à noir sont toujours exclusifs et tous les montages extérieurs concernant ces signaux sont possibles.

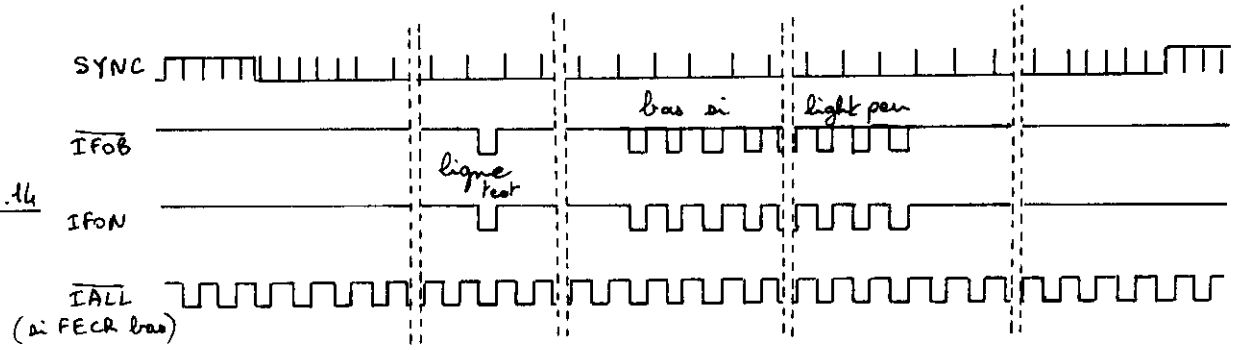


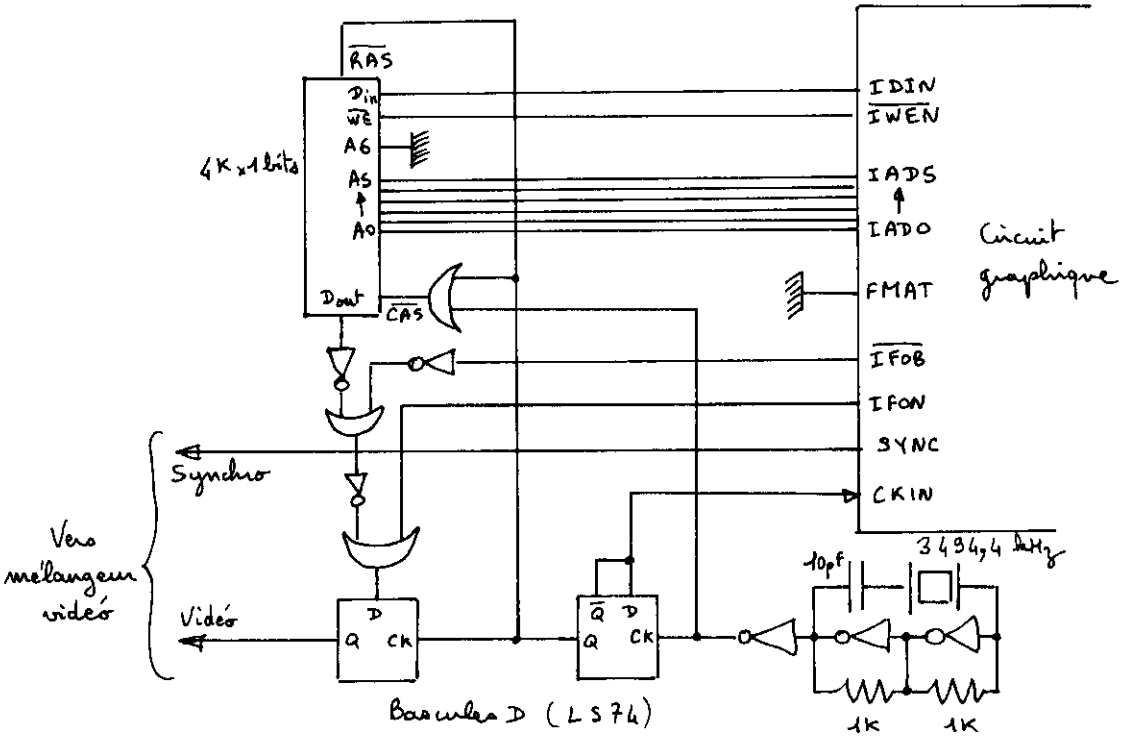
Fig 2.14

2.6-EXEMPLES DE MONTAGES (NOIR ET BLANC)

a)64x64. Un seul boîtier 4Kx1 bits suffit. Les bits sont accédés séquentiellement le long d'une ligne. Chaque ligne est répétée 4 fois. Il est nécessaire de disposer d'une horloge au double de la fréquence point pour générer $\overline{CAS}$. La sortie de la mémoire n'étant pas toujours valide, une bascule est nécessaire. Dans cette application, nous obtenons un affichage graphique 64x64 N&B en 5 boîtiers (Fig.2.15).

b)128x128. Il faut 2 boîtiers de 8Kbits (INTEL 2108 par ex.), ou 4 boîtiers de 4Kbits. IAD6 n'est utilisé que pour sa partie basse.

Fig 2.15 : Application 64x64 Noir et Blanc



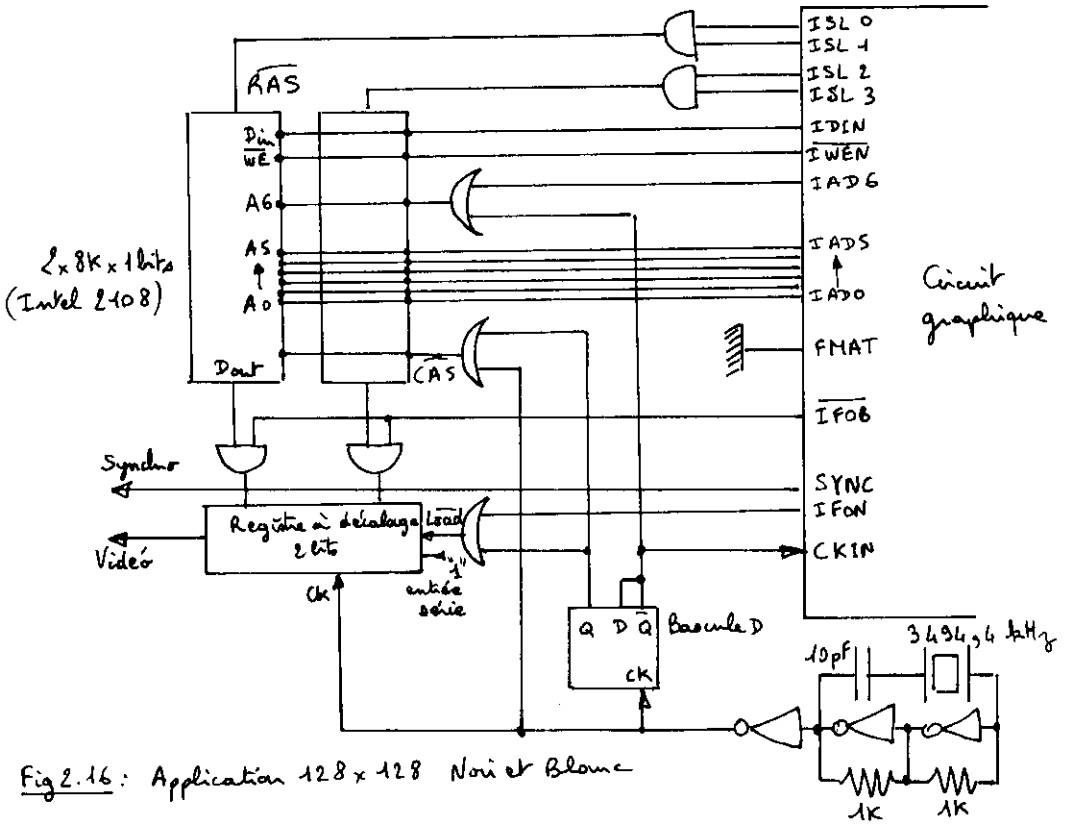


Fig 2.16 : Application 128 x 128 Noir et Blanc

Une horloge à la fréquence point décale le registre et permet de générer $\overline{CAS}$. IFON intervient en empêchant le chargement du registre. Dans ces conditions, il y a bien forçage à noir si l'entrée série du registre est au niveau haut. On peut utiliser une seule mémoire 16K si elle a un temps de cycle inférieur à 275 ns. On fait alors des accès à la fréquence points. L'adresse supplémentaire est fournie en écriture par les ISL, et en lecture par le diviseur extérieur.

c)<u>256x256</u>. On utilise 4 boîtiers 16Kbits.

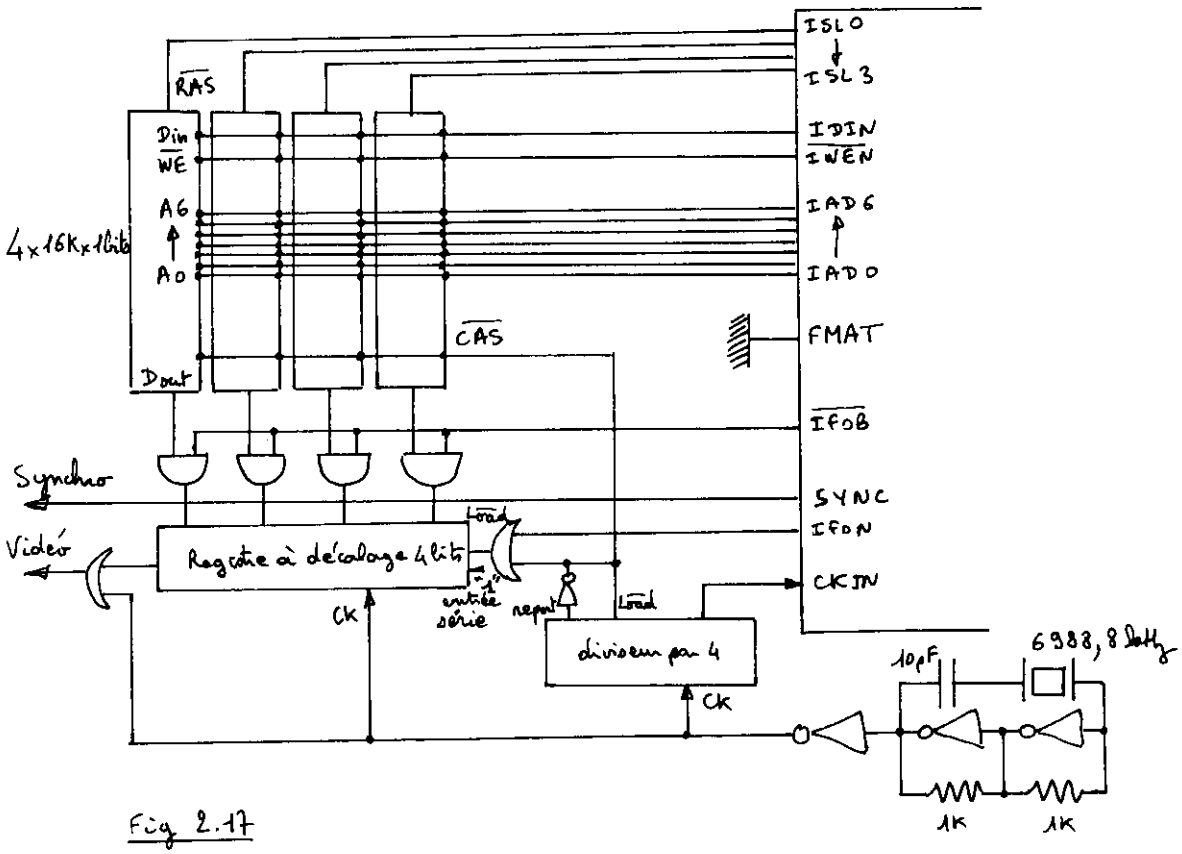


Fig 2.17

d)<u>512x512</u>. Les 4 sorties ISLi doivent être décodées pour commander les 16 mémoires 16K. Celles-ci sont arrangées en 2 moitiés de 8 boîtiers. Lors de la lecture, une moitié entière est sélectée. Ceci se fait en conjuguant $\overline{IALL}$ et ISL3 qui porte alors une adresse de visualisation.

L'ensemble de toutes les cases mémoire est lu sur 2 trames TV.

La séparation des lignes TV suivant leur parité ne coïncide pas avec la séparation en 2 moitiés, sinon une moitié ne serait pas rafraîchie pendant une trame. Il y a basculement de la moitié utilisée toutes les 2 lignes d'une même trame(grâce à la sortie ISL3), soit tous les 128 accès.

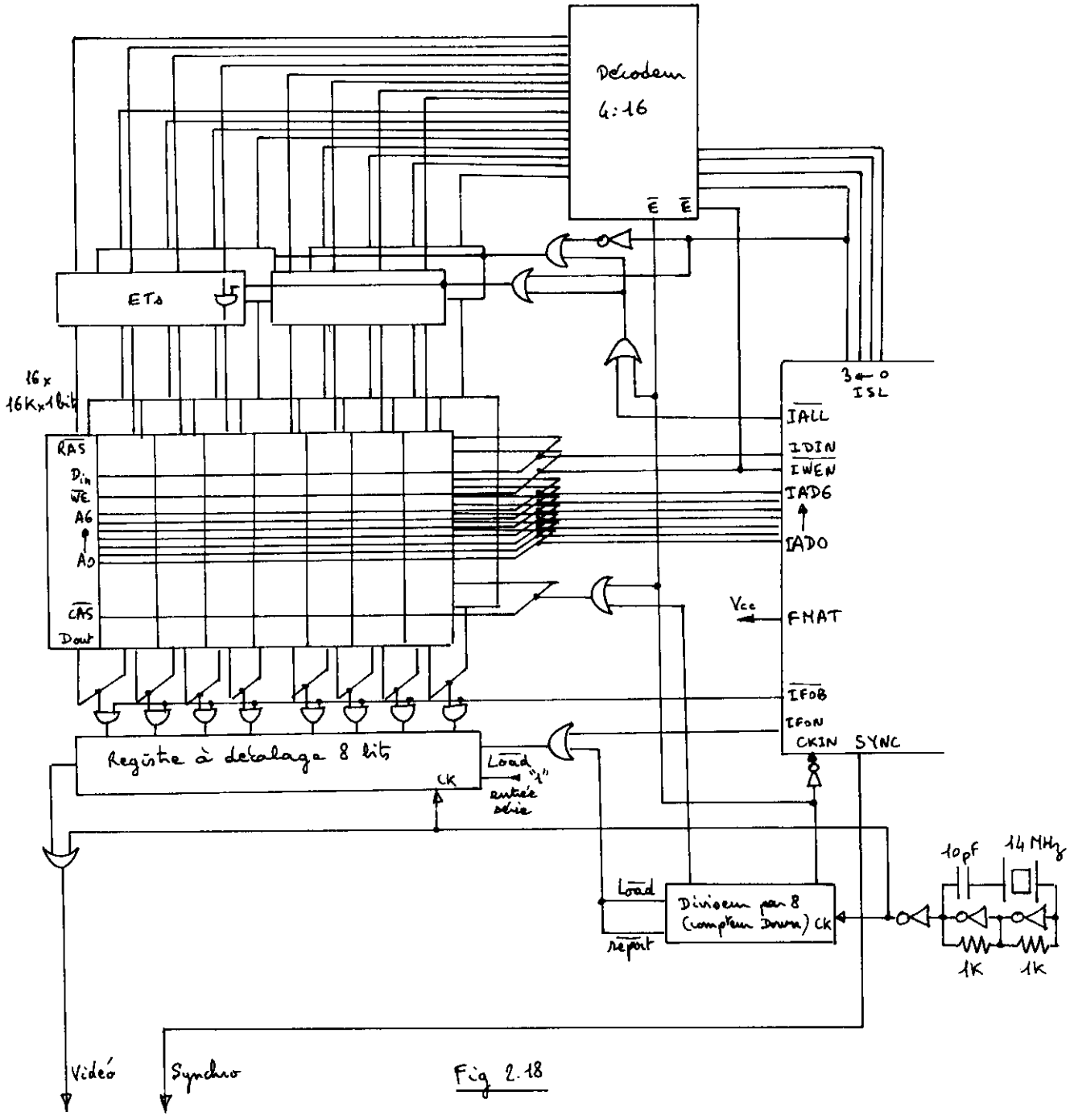


Fig 2.18

e)Autres formats. Vu le mode de sélection des boîtiers par les sorties ISL, il n'est pas possible d'avoir un format dont le nombre de colonnes ne soit pas une puissance de 2. De même, il est difficile d'utiliser le boîtier pour une définition verticale qui ne soit pas non plus une puissance de 2 .

Par contre, il n'est pas obligatoire d'avoir des définitions horizontale et verticale égales, mais cela présente peu d'intérêt. Des définitions égales ne signifient pas un écran carré. On peut en effet régler les amplificateurs de déviation du moniteur pour déformer l'image. Mais alors, une composante des vecteurs et des caractères devra subir une homothétie software pour corriger la déformation du dessin.

2.7-SIGNAL DE SYNCHRONISATION

Il est généré sur la sortie SYNC. Sa forme est celle de la figure 2.5, la durée des paliers étant 8T ou 112-8=104T, si T est la période de l'horloge CKIN . Ce signal ne change d'état que sur un front descendant de CKIN puisqu'il est généré à partir de reconnaisseurs sur le compteur de visualisation. Dans le cas où FMAT est bas, tous les tops trames sont ceux de la figure 2.5.b.

Ce circuit peut facilement synchroniser une autre source de signal vidéo puisque les signaux de cadrage peuvent se déduire de SYNC, $\overline{\text{IALL}}$,$\overline{\text{IFOB}}$, et IFON . Par contre, il ne comprend pas les fonctions nécessaires pour se synchroniser lui-même sur une autre source. Toutefois, il est possible de le faire en agissant sur l' horloge CKIN qui peut très bien voir l'une quelconque de ses phases s'allonger.

2.8-RECAPITULATION DE LA PARTIE VISUALISATION

On peut représenter toute la partie lecture-synchronisation-gestion mémoire de la façon suivante:

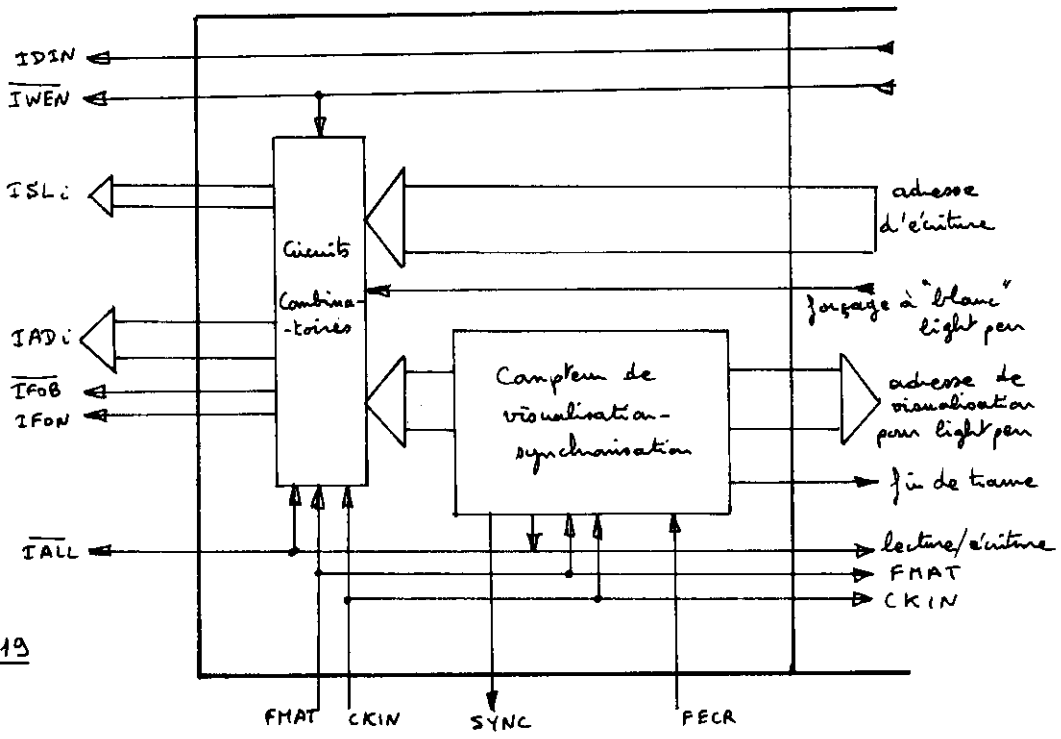


Fig 2.19

Tous les signaux à droite sont des signaux d'échange avec la circuiterie d'écriture. L'entrée FECR ("forçage en écriture") supprime toutes les phases de visualisation, et laisse 112 périodes par ligne pour l'écriture. IFON et $\overline{\text{IALL}}$ sont alors toujours à l'état haut.

Le signal "fin trame" est haut de la dernière ligne visualisée d'une trame à la première ligne de la trame suivante. Il est utilisé par le système d'écriture dans la gestion du light-pen, de l'effacement de l'image, et peut être la cause d'une interruption (§2.15).

2.9-SYSTEME D'ECRITURE

Cette partie du circuit doit être considérée comme une interface entre la partie visualisation et le bus microprocesseur.

Elle est entièrement sous le contrôle de la visualisation par les signaux CKIN et $\overline{IALL}$. Son rôle est de fournir à chaque cycle mémoire d'écriture une adresse et le signal $\overline{IWEN}$.

Du côté bus microprocesseur, elle simule une suite de 13 cases mémoires qui peuvent être accédées par le microprocesseur en lecture ou en écriture à l'aide d'un bus de données bidirectionnel de 8 bits (MPDi) et d'un bus d'adresse de 4 bits (MADi). Les échanges sont contrôlés par un signal temporel ($\overline{MPCE}$) et un signal de lecture/ écriture (MR/$\overline{W}$). Certaines cases correspondent à des registres internes (tel que l'adresse d'écriture X,Y par exemple). D'autres cases n'ont pas la même signification en lecture et en écriture. La case 0 par exemple est l'endroit où l'on écrit les commandes, et où on peut lire le mot d'état. La description précise de ces registres est donnée §2.18. (voir aussi figure 11.1)

2.10-REGISTRES X ET Y

L'adresse d'écriture est constituée de 2 registres de 12 bits chacun: X et Y. Ils permettent donc d'adresser un espace de 4096x4096 points. Ils donnent en permanence l' adresse où pourrait s'effectuer une écriture en mémoire, équivalente à l'adresse du "spot" d'une visu à balayage cavalier. Cette adresse d'écriture peut être modifiée (ce spot peut être déplacé) de 2 façons: ou bien on la fixe de manière absolue par un chargement des registres X,Y; ou bien on utilise le générateur de vecteurs et de caractères qui au cours d'un traçage incrémente ou décrémente les registres X ou Y. Ces 2 registres de 12

bits sont chacun découpés en une partie basse de 8 bits et une partie haute de 4 bits. Cette dernière est étendue à 8 bits avec des zéros placés devant. Ce découpage permet leur accès par le bus 8 bits du microprocesseur (adresses 4,5,6,7). (voir tableau 2.1)

Parmi les 4096x4096 points adressables, seule une fenêtre est visualisée et réellement mémorisée. Si le spot est en dehors de cette fenêtre, l'écriture est inhibée (signal $\overline{IWEN}$ forcé haut). L'espace adressable est cyclique, c'est-à-dire que si le spot dépasse sur un bord, il apparaît sur le bord opposé. Ceci, associé à une description relative des vecteurs (par leurs projections ΔX et ΔY) et des caractères, permet de résoudre automatiquement la plupart des problèmes de découpage d'une fenêtre dans une figure. Toute portion d'une grande figure4096x4096 est visualisable. Il suffit de déplacer initiallement, non pas la fenêtre, mais la grande figure par un seul positionnement du spot. Comme tous les éléments sont définis de façon relative, la figure sera globalement translatée:

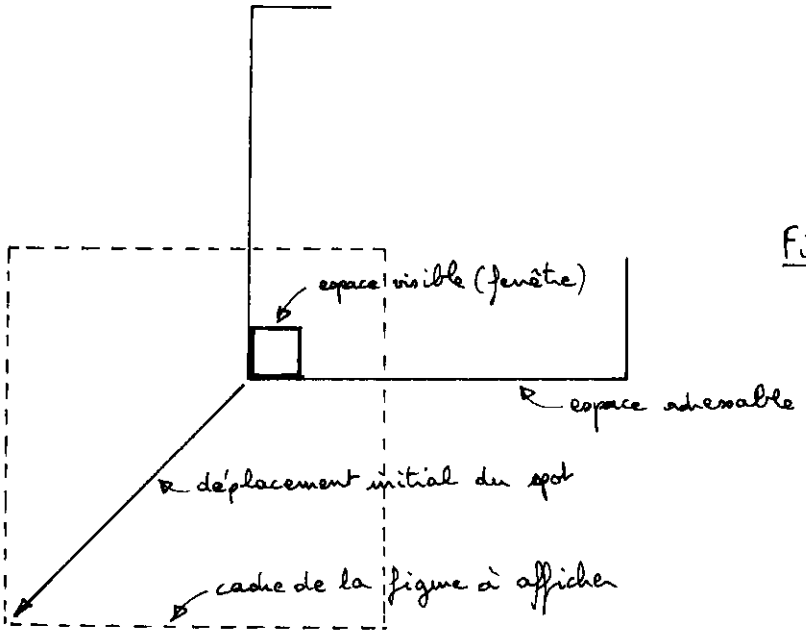


Fig 2.20

Néanmoins, il est possible de restreindre l'espace cyclique à celui de la fenêtre, en positionnant le bit de poids fort du registre de contrôle (adresse 1 -voir tableau 2.1). Quelquesoit la valeur de ce bit, il est en outre possible de savoir si l'adresse (complète)

d'écriture correspond à un spot hors de la fenêtre ou non (bit 3 du mot d'état). Il est à noter que cette dernière information est fonction de l'entrée FMAT.

Indépendemment de la position du spot, celui-ci peut être "allumé" ou "éteint" (s'il est éteint, le signal $\overline{IWEN}$ est toujours haut). Il peut aussi être "marquant" ou "effaçant" (cette caractéristique commande la sortie IDIN). Ces 2 caractères sont contrôlés par les bits 2 et 3 du mot de contrôle.

2.11-GENERATEUR DE VECTEURS

C'est un automate qui reçoit en entrée les projections du vecteur ΔX et ΔY, et qui génère en sortie des incréments (ou décréments) pour les registres X et Y.

On spécifie séparément $|\Delta X|$, $|\Delta Y|$ et les signes associés. $|\Delta X|$ et $|\Delta Y|$ sont 2 registres de 8 bits accessibles par le bus microprocesseur (adresses 2 et 3), ce qui impose:

$$\begin{cases} -255 \leqslant \Delta X \leqslant 255 \\ -255 \leqslant \Delta Y \leqslant 255. \end{cases}$$

Les signes (et nullités éventuelles) sont spécifiés par les 3 bits de poids faibles d'un octet de commande de la façon suivante: (codes de commande H'10' à H'17'+)

bits faibles:

000	$\Delta X \geqslant 0$; $\Delta Y = 0$
001	$\Delta X \geqslant 0$; $\Delta Y \geqslant 0$
010	$\Delta X = 0$; $\Delta Y \geqslant 0$
011	$\Delta X \leqslant 0$; $\Delta Y \geqslant 0$
100	$\Delta x \leqslant 0$; $\Delta Y = 0$

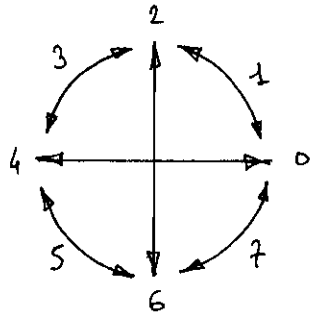


Fig 2.21

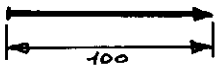
+ H est mis pour hexadécimal.

101	$\Delta X < 0$; $\Delta Y \leqslant 0$
110	$\Delta X = 0$; $\Delta Y < 0$
111	$\Delta X > 0$; $\Delta Y < 0$

Ceci permet de tracer de nombreuses figures géométriques simples en changeant peu souvent les registres $|\Delta X|$ et $|\Delta Y|$, et donc à raison de 1 octet par vecteur. Pour le cas où les vecteurs tracés sont parallèles aux axes ou aux bissectrices, des codes de commande spéciaux (H'18' à H'1F') permettant de ne tenir compte que d'un seul registre $|\Delta X|$ ou $|\Delta Y|$. Les projections des vecteurs sont alors choisies égales dans les 2 directions à:

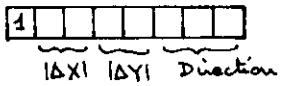
$$|\Delta X| = |\Delta Y| = \sup(\text{registre } |\Delta X| , \text{ registre } |\Delta Y|)$$

Les signes et nullités sont spécifiés de la même façon que pour les autres vecteurs.

Remarque: dans le cas $|\Delta X|=50$ et $|\Delta Y|=100$, le code de commande H'18' (direction 0) trace le vecteur: 

Ces codes sont nommés "Directions privilégiées" dans le tableau 2.2.

Pour les cas de tracés de petits vecteurs ($|\Delta X| \leqslant 3$ et $|\Delta Y| \leqslant 3$), des codes de commande spéciaux (H'80' à H'FF') spécifient en un seul octet ΔX et ΔY d'après le format suivant:



et sans changer les registres $|\Delta X|$ et $|\Delta Y|$ qui peuvent être utilisés conjointement pour de grands vecteurs. Tous ces codes spéciaux permettent de minimiser l'information nécessaire pour décrire une image composée de traits.

Les vecteurs peuvent être tracés de 4 types de traits différents spécifiés par les 2 bits de poids faibles du registre de contrôle: trait continu, trait pointillé (1 point allumé, 1 point éteind), trait tireté (2 points allumés, 2 points éteinds), et trait mixte

(10 points allumés, 2 éteinds, 2 allumés, 2 éteinds). Le circuit générant les traits non continus est toujours positionné de la même façon pour un vecteur donné au départ du traçage. Ceci garantit que l'effacement d'un vecteur non continu puisse s'obtenir en ne changeant que la bascule Marquant/Effaçant, à condition toutefois que le vecteur soit parcouru dans le même sens qu'au traçage.

Lors d'un tracé, le générateur fait passer le spot de l'adresse (X_o, Y_o) à l'adresse $(X_o+\Delta X, Y_o+\Delta Y)$. Le temps moyen de tracé est environ $1,3(m+3)\mu s$ (si m est $\sup(|\Delta X|, |\Delta Y|)$) ou $0,55(m+3)\mu s$ si on utilise le forçage en écriture (broche FECR). Le nombre d'accès mémoire pour ce tracé est m+1, car les points origine et extrémité sont inscrits. Ainsi, si le spot est allumé, le traçage d'un vecteur pour lequel $\Delta X=\Delta Y=0$ a pour effet de marquer le point où se trouve le spot.

Le générateur de vecteurs peut se symboliser ainsi:

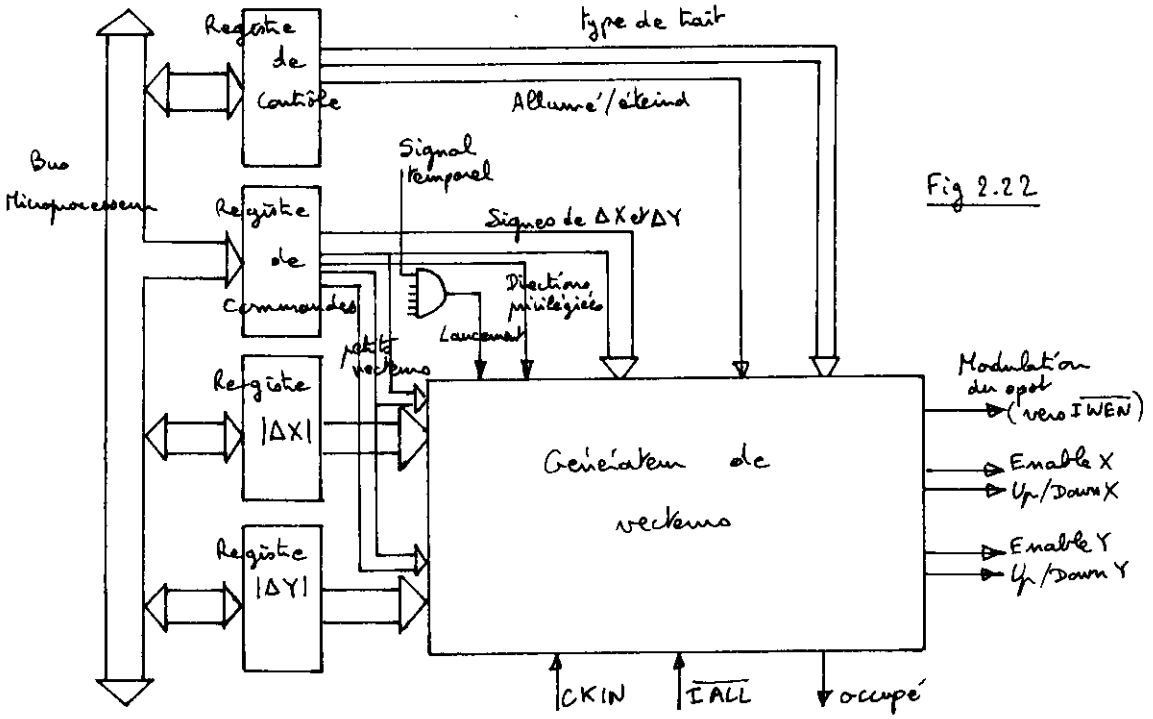


Fig 2.22

2.12-GENERATEUR DE CARACTERES

Le générateur de caractères est comparable au générateur de

vecteurs en ce qui concerne son branchement. Il génère des incréments et des décréments pour X et Y. Ses entrées sont d'une part le code du caractère à tracer (parmi les 95 caractères full-ASCII imprimables), et d'autre part 2 facteurs d'homothétie à appliquer à ce caractère suivant x et suivant y. Ces 2 facteurs peuvent varier indépendemment et de 1 à 16. Les caractères sont tracés à partir d'une matrice 5x7, et l'espace entre 2 caractères est de 1 colonne. Le spot lors du traçage passe de (X_o,Y_o) à (X_o+6p,Y_o). Ceci permet de tracer jusqu'à 64 lignes de 85 caractères (dans le format 512x512).

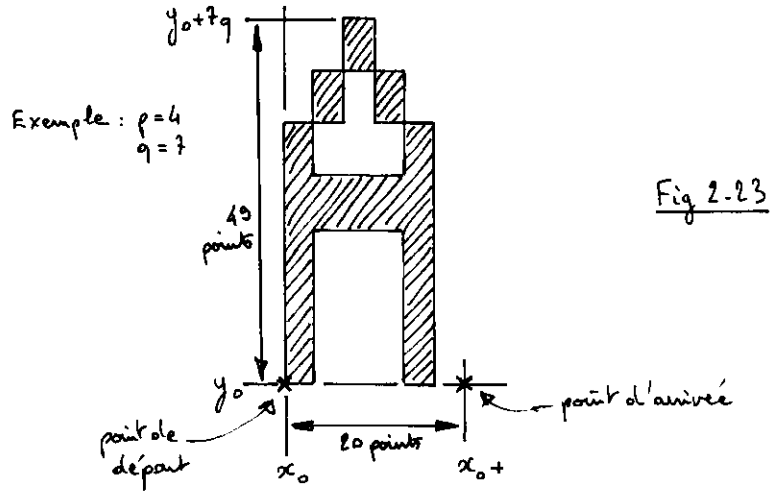


Fig 2.23

A partir du même générateur, on peut tracer des "pavés" de 2 sortes:

-un pavé de 5px7q points allumés qui permet aussi d'effacer tout caractère,

-un pavé de 4px4q points pour lequel le spot passe de (X_o,Y_o) à (X_o+4p,Y_o).

Les facteurs p et q sont accessibles par le bus microprocesseur où ils se nomment "tailleXcar" et "tailleYcar" (adresses H'8' et H'9'). Ils sont constitués de 4 bits étendus à 8 avec des zéros placés devant. La valeur H'00' donne un facteur égal à 16. Les quatre bits forts sont toujours ignorés en écriture, nuls en lecture.

Le générateur de caractères peut se schématiser ainsi:

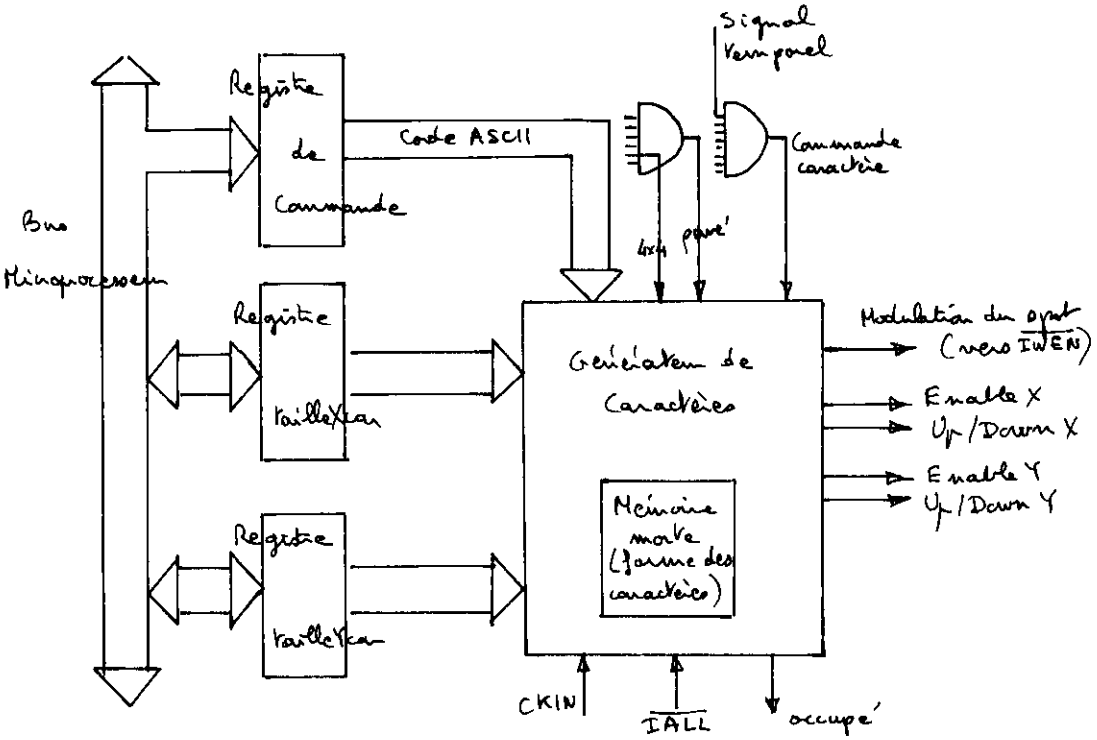


Fig 2.14

2.13-LIGHT-PEN ET RETICULE

Si le circuit a reçu le code de commande "armement light-pen" (code H'08') ou "armement réticule" (H'09'), il attend lors de la trame suivante un front montant sur l'entrée LPEN qui sert à échantillonner la valeur courante de l'adresse de visualisation dans 2 registres "Xlight-pen" et "Ylight-pen". Ces registres sont accessibles en lecture seule sur le bus microprocesseur (adresses H'B' et H'C').

Dans le cas d'un armement light-pen, la vidéo est "forcée à blanc" (par la broche $\overline{IFOB}$) lors de la trame considérée. Dans le cas du réticule, les traits sur l'image doivent être générés extérieurement au boîtier. Le OU logique des 2 signaux "trait vertical" et "trait horizontal" doit être envoyé sur la broche LPEN du circuit (Il est souvent préférable de dessiner le réticule par programme. Pour éviter d'effacer l'image, il peut être tracé par complémentation du contenu [10], en faisant fonctionner la mémoire en "Read-Modify-Write").

X light pen est l'adresse d'un mot de la mémoire d'image le long d'une ligne. Seuls ses 6 bits de poids forts sont significatifs.

En format 512x512, Xlight pen et Y light pen sont toujours cadrés sur un octet, comme en 256x256. Il faut alors les multiplier par 2 pour obtenir l'adresse logique réelle si nécessaire.

Il faut soustraire une quantité constante au registre Xlight pen pour compenser les retards introduits entre la sortie d'une adresse sur IADi et le signal LPEN. Ces retards dépendent du montage externe.

Les signaux internes de gestion du light pen sont représentés en figure 2.25. Dans le cas du réticule, la seule différence est l'absence de "forçage à blanc".

Le signal "light pen non lu" est accessible en bit faible du registre X light pen. Il est remis à 0 par la lecture du registre X light pen ou de Y light pen. Le signal "light pen terminé" est accessible dans le mot d'état et peut être à l'origine d'une interruption.

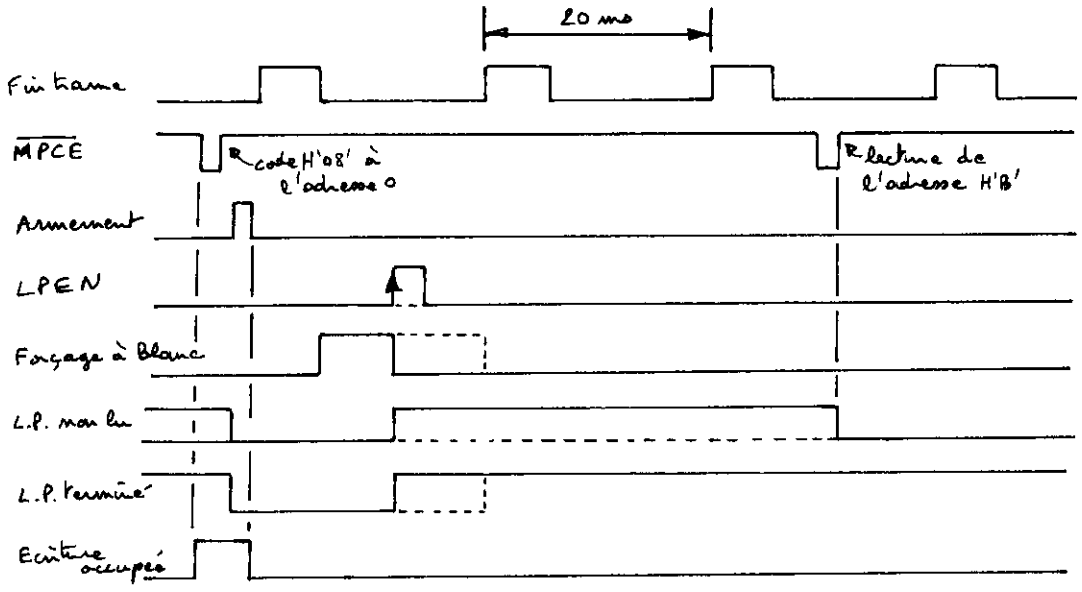


Fig 2.25

2.14-MECANISME D'EFFACEMENT DE L'IMAGE ENTIERE

Pour effacer toute l'image, ou pour la positionner à un état particulier (fond non Noir), 2 codes spéciaux sont prévus (H'04' et

H'0C'). Ils agissent pendant les phases de visualisation. Le temps minimal nécessaire est donc une trame si FMAT est bas, 2 trames si FMAT est haut. Ces trames sont comptées à partir du front descendant de "fin trame" qui suit la commande. Les signaux de contrôle sont représentés en figure 2.26. Le signal effacement force $\overline{IWEN}$ bas. Dans le cas du code H'04', il force aussi IDIN haut. L'action de ces codes est indépendante des bits 2 et 3 du registre de contrôle, et ne les modifie pas.

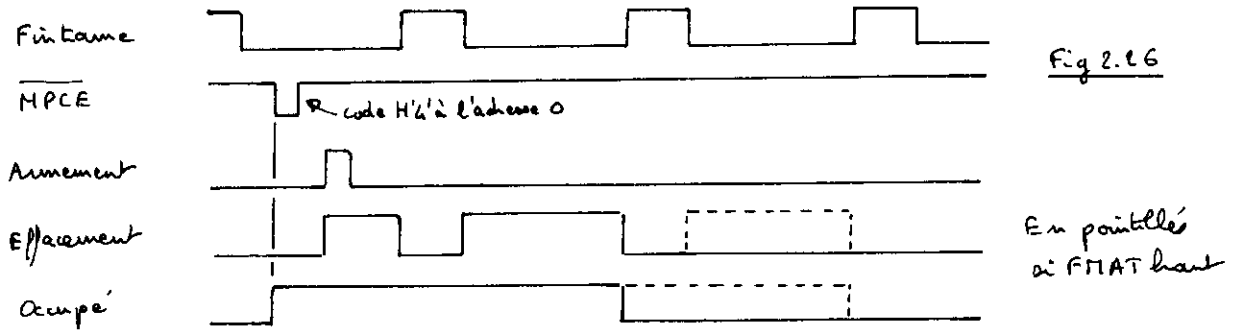


Fig 2.26

Si les boîtiers mémoire sont montés pour fonctionner en "Read-Modify-Write" lors de la visualisation, la(les) trame(s) effaçante(s) sera(ont) visualisée(s). Ceci permet de faire du dessin animé à une vitesse maximale, en écrivant le nouveau dessin lorsque "fin trame" est haut, et en lançant l'effacement avant même le début de la trame. La visualisation effacera la mémoire, prête pour la trame suivante. Dans une telle utilisation, il peut être souhaitable d'augmenter la vitesse moyenne d'écriture pour augmenter le nombre de points inscriptibles lors de "fin trame". Ceci peut être obtenu par un montage permettant de commander l'entrée FECR par programme. Mais ceci est dangereux car supprime le rafraîchissement.

2.15-MECANISME D'INTERRUPTION

Trois signaux internes différents peuvent être à l'origine d'une

interruption:

-"système d'écriture libre": Ce signal est le NOR de 4 signaux: "générateur de vecteurs occupé", "générateur de caractères occupé", "effacement occupé", et "analyseur de commandes occupé". Il doit être testé avant l'envoi de toute commande pour ne pas perturber l'exécution de la précédente.

-"fin trame": Ce signal peut servir d'horloge à 50Hz (dont la précision est liée à celle de l'horloge CKIN), ou mieux permet de synchroniser une modification de l'image avec le défilement des trames.

-"light pen terminé".

Le système de prise en compte des interruptions est le suivant:

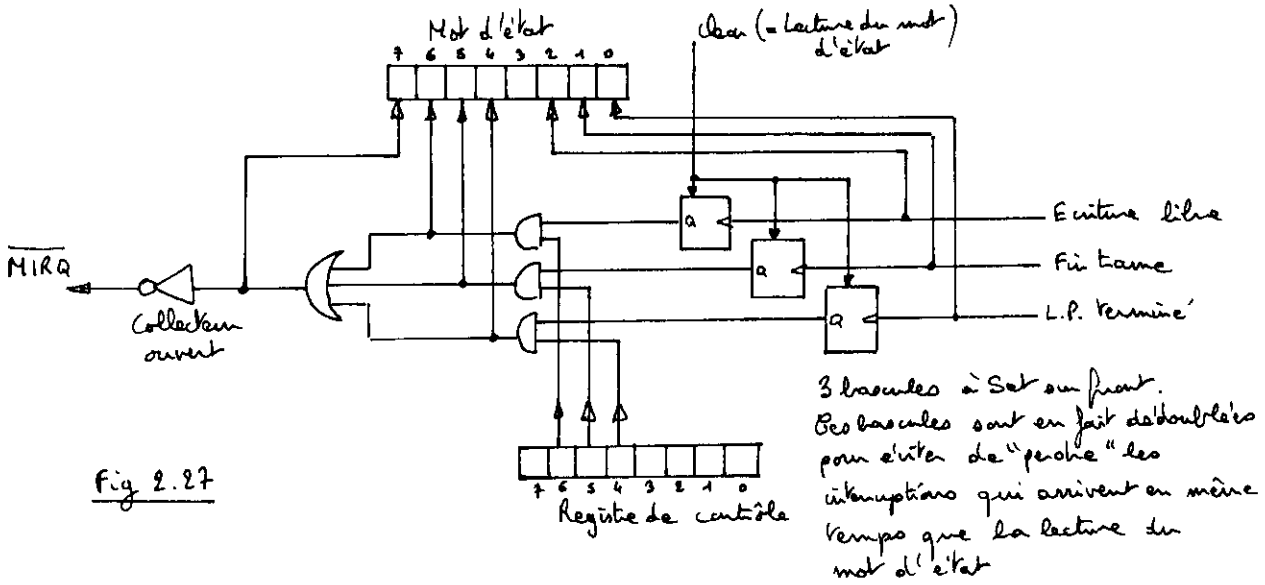


Fig 2.27

2.16-LECTURE D'UN POINT DE LA MEMOIRE D'IMAGE

Cette commande (code H'OF') déclenche le mécanisme d'écriture au point adressé par X,Y sans modifier cette adresse. Mais, au moment de l'écriture, au lieu de baisser $\overline{\text{IWEN}}$, le circuit baisse une autre sortie: $\overline{\text{LSTR}}$. C'est donc une lecture à l'adresse X,Y qui a lieu. La sortie des mémoires lors de cette lecture peut être échantillonnée

par $\overline{\text{LSTR}}$ dans un registre accessible extérieurement par le microprocesseur. De cette manière, le programmeur peut travailler point par point dans la mémoire d'image. Suivant le montage, on recueille un mot entier (dont la longueur dépend de la configuration mémoire) si $\overline{\text{LSTR}}$ force $\overline{\text{IALL}}$, ou un seul point si on se sert de la sélection par ISL (Fig.2.30 et 2.28 respectivement).

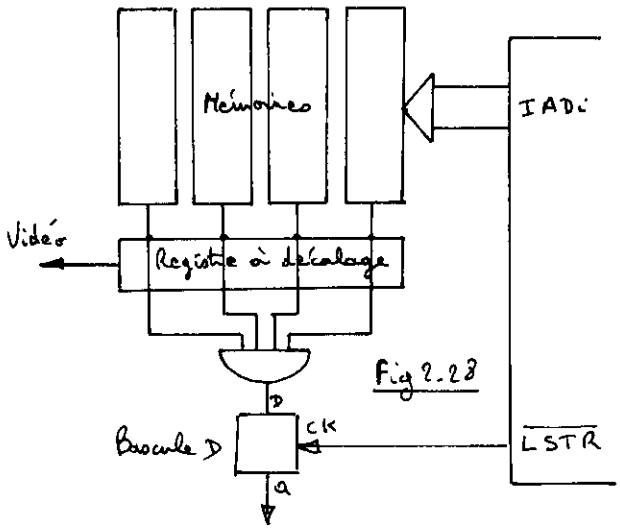


Fig 2.28

Exemple: lecture d'un point en 256x256 N&B. On a utilisé des mémoires dont la sortie est haute quand elles ne sont pas sélectées.

2.17-LIAISON AU BUS MICROPROCESSEUR

Les signaux d'échange avec le microprocesseur sont représentés Fig.2.31. Les adresses des registres et les codes de commande sont résumés dans les tableaux 2.1 et 2.2. Les codes H'0', H'1', H'2', H'3' permettent de modifier séparément les bits du mot de contrôle concernant le spot. Les codes H'5', H'6', H'7', H'D', H'E' permettent de remettre à zéro les registres X,Y, ensemble ou séparément. Les codes H'4', H'6', H'7', H'C' effacent l'écran. Le code H'7' réinitialise en outre le circuit en remettant à zéro les registres: Contrôle, $|\Delta X|$, $|\Delta Y|$, et en positionnant à "un" les registres tailleXcar et tailleYcar.

Le mode d'accès aux registres a été choisi pour pouvoir accéder à tout ce qui définit l'état de la visu. Cela permet d'avoir plusieurs programmes fonctionnant à des niveaux d'interruption différents,

chaque changement de niveau étant associé à une sauvegarde ou une restauration des registres du circuit.

La case mémoire d'adresse H'A' a pour seul effet de générer une réplique de $\overline{MPCE}$ sur la broche $\overline{ISTR}$ (quelquesoit le niveau de MR/$\overline{W}$). Son intérêt est de minimiser la circuiterie externe si on a besoin d'un registre supplémentaire (registre de couleur, ou registre de lecture sur $\overline{LSTR}$). La sortie $\overline{ISTR}$ peut être utilisée uniquement en lecture, ou uniquement en écriture ou dans les 2 sens. Deux exemples d'utilisation sont donnés en Fig.2.29 et 2.30. Dans le deuxième exemple, il est entendu que le programmeur n'utilise cette adresse qu'en lecture. Pour plus de sécurité, on peut ajouter la condition MR/$\overline{W}$ sur la commande de lecture du registre.

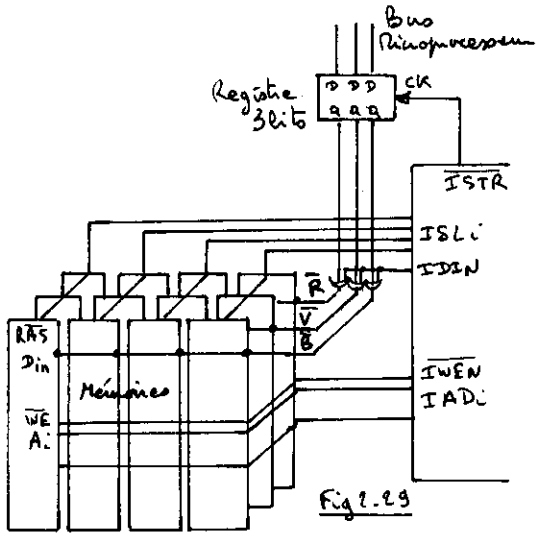


Fig 2.29

Utilisation de la sortie $\overline{ISTR}$ en écriture pour échantillonner un registre de couleur dans une application 256x256points 8 couleurs.

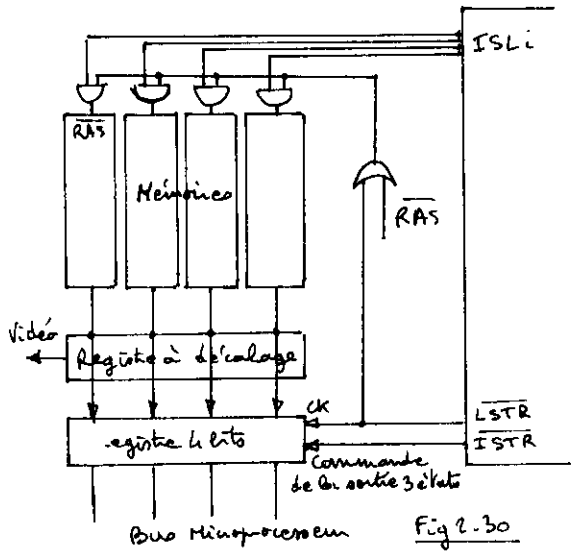
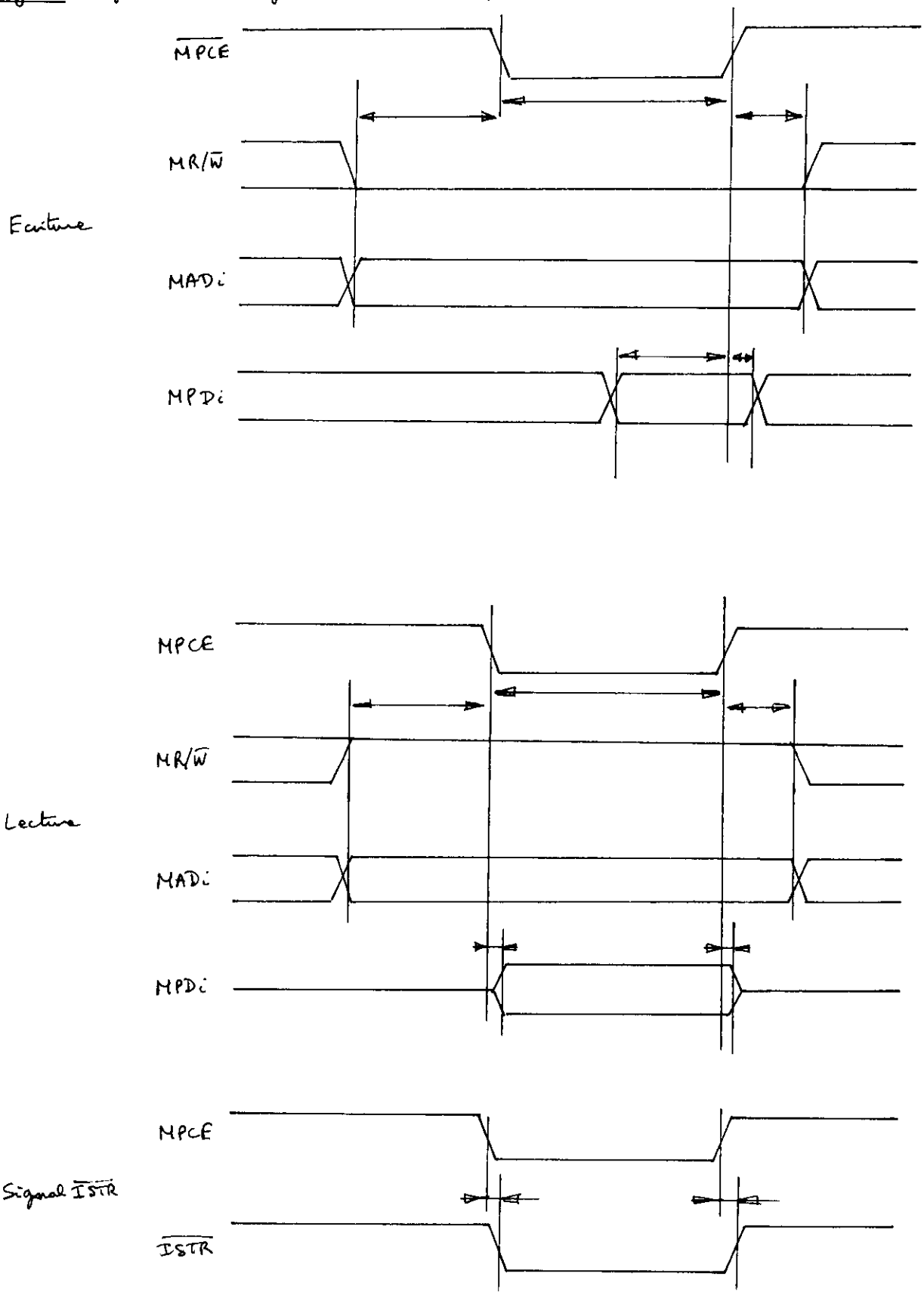


Fig 2.30

Utilisation de la sortie $\overline{ISTR}$ en lecture pour positionner sur le bus microprocesseur la valeur lue après l'utilisation de $\overline{LSTR}$.
Remarque: on sait à quel moment la lecture a été effectuée par la

surveillance du signal "libre" (bits 2 et 6 du mot d'état).

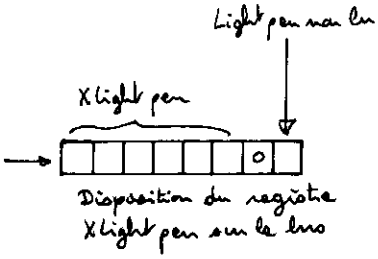
Fig 2.31 : Signaux de dialogue avec le microprocesseur



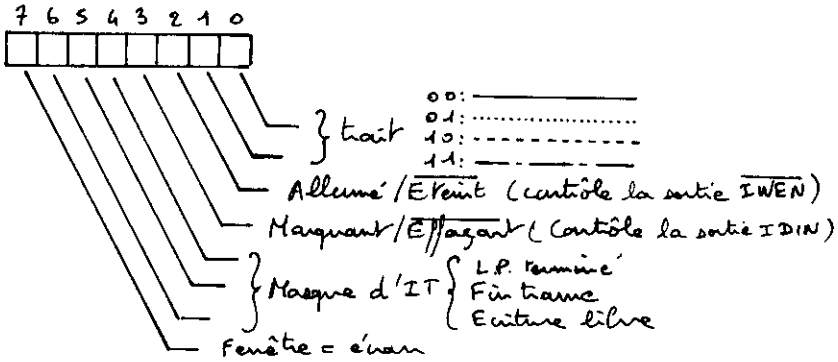
2.18-RECAPITULATION DES CODES

Adresses

N°	Ecriture	Lecture
0	Commande	Mot d'état
1	Contôle	Contôle
2	|ΔX|	|ΔX|
3	|ΔY|	|ΔY|
4	X haut	X haut
5	X bas	X bas
6	Y haut	Y haut
7	Y bas	Y bas
8	taille X car	taille X car
9	taille Y car	taille Y car
A	Registre externe	Registre externe
B		X Light pen
C		Y light pen



Mot de contôle



Mot d'état

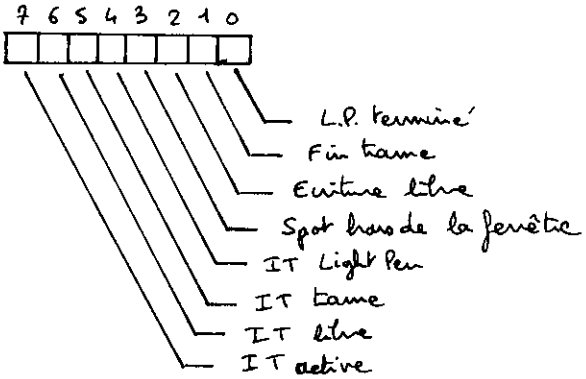


Tableau 2.1

3 2 1 0 \ 7 6 5 4		0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1	1 0 0 0	1 0 0 1	1 0 1 0	1 0 1 1	1 1 0 0	1 1 0 1	1 1 1 0	1 1 1 1
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0 0 0 0	0	Mode Marqt	Traçages Vecteurs	espace	0	@	P	`	p	petits vecteurs							
0 0 0 1	1	Mode Effct		!	1	A	Q	a	q								
0 0 1 0	2	Spot Allumé		"	2	B	R	b	r								
0 0 1 1	3	Spot Eteint		#	3	C	S	c	s								
0 1 0 0	4	Effact total		$	4	D	T	d	t								
0 1 0 1	5	Positiont en (0,0)		%	5	E	U	e	u								
0 1 1 0	6	Positiont d'Effct		&	6	F	V	f	v								
0 1 1 1	7	Reinit.		'	7	G	W	g	w								
1 0 0 0	8	Avant L.P.	Vecteurs de directions privilégiées	(	8	H	X	h	x								
1 0 0 1	9	Réticule		)	9	I	Y	i	y								
1 0 1 0	A	Pavé 5x7		*	:	J	Z	j	z								
1 0 1 1	B	Pavé 4x4		+	;	K	[	k	{								
1 1 0 0	C	Fond		,	<	L	\	l	|								
1 1 0 1	D	Clear X		-	=	M	]	m	}								
1 1 1 0	E	Clear Y		.	>	N	↑	n	~								
1 1 1 1	F	Lecture points		/	?	O	←	o	Nop								

petits vecteurs

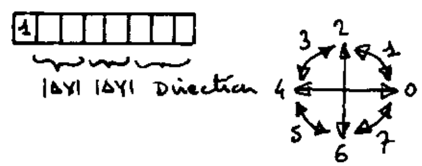


Tableau 2.2

3-METHODE D'IMPLANTATION

d)Bascule D-D'.

e)Bascule D avec Clear et Set asynchrones.

f)Bascule à Set sur front et Clear asynchrone.

g)Bascule D avec Enable.

h)Bascule T ou $\bar{T}$ avec Clear ou/et Set synchrone.

i)Bascule T ou $\bar{T}$ avec Load synchrone.

j)Report de compteur UP.

k)Report de compteur UP/DOWN.

l)Additionneur.

m)Comparateur d'égalité.

3.10-Plan des chapitres suivants.

3.1-INTRODUCTION

Le chapitre 2 décrit le circuit intégré graphique en tant que "boîte noire", dans le langage des utilisateurs: concepteurs de consoles et programmeurs. Il faut maintenant passer à une description détaillée en vue de la réalisation physique du circuit. C'est le dessin des "masques"; toutes les étapes ultérieures de la réalisation (découpage du stabilène, réduction à l'échelle, photorépétition et traitement chimique) sont pratiquement automatisées, et en tous cas ne font pas partie de la conception proprement dite.

Le passage de la description fonctionnelle du circuit au plan des masques peut se comparer à la suite: compilation, édition de liens, et chargement pour l'exécution d'un programme écrit en langage évolué. Mais dans notre cas, aucune de ces étapes n'est automatisée, et ce travail demande plusieurs mois.

Pour des circuits d'une faible complexité (quelques centaines de transistors), ou pour ceux dont le taux de répétitivité est élevé,

on peut commencer par la mise au point d'un schéma électrique décrivant le circuit au niveau des transistors et des connexions. Le fonctionnement est alors testé à l'aide de "simulateurs électriques", programmes appliquant les lois de KIRCHHOFF pour ce réseau.

Mais la complexité atteinte aujourd'hui couramment pour les circuits commercialisés est 10.000 transistors (100.000 pour des circuits hautement répétitifs tels que les mémoires). Or, tous les circuits de cette complexité qu'il serait intéressant d'intégrer ne sont pas forcément répétitifs!! On peut toujours dégager de petits sous-ensembles qui le sont, mais il faut une fonction vraiment spéciale pour que tout le circuit le soit.

Passer directement au dessin des masques est impossible car cela demande la manipulation d'une quantité d'information gigantesque. D'autant plus que, la correction des erreurs ne pouvant alors s' effectuer qu'après la réalisation physique du circuit, ce travail est comparable à l'écriture en binaire d'un programme de 100.000 instructions dont le chargement avant exécution pour test prendraient 6 mois à chaque fois.

Or, l'automatisation de ce travail est beaucoup plus compliquée que l'écriture d'un compilateur pour un langage de haut niveau. En effet, ce "compilateur" devrait manipuler une information qui n' apparaît pas dans le langage source: les connexions. Pour un programme, toutes les références à une variable sont identiques car on part du principe que le temps d'accès à toutes les cases de la mémoire est identique. Mais dans le dessin des masques d'un circuit, une connexion dépend de la localisation des variables. Elle occupe une certaine surface comme un opérateur, et pose des problèmes de topologie [3] . Le "compilateur" doit donc créer ces nouveaux éléments dont la manipulation est souvent plus délicate que celle des

opérateurs décrits dans le langage source. Pour ces raisons, de tels compilateurs n'existent pas encore, et ce travail doit se faire manuellement.

La mise au point de la méthode a donc une importance capitale. En particulier, il est indispensable de pouvoir valider la logique dans une des toutes premières étapes(ce qui est équivalent à la "preuve" d'un programme écrit dans un langage de haut niveau). De plus, la chaîne de transformation entre cette étape validée et le dessin final doit elle-même être prouvée. Ce n'est qu'à ces conditions que l'on peut être sûr que des mois de travail ne seront pas à recommencer.

3.2-METHODE UTILISEE

Nous avons choisi de progresser dans la définition du circuit de la façon suivante (inspirée du chemin utilisé par J.GASTINEL pour le dessin du circuit SFF 96364 [8]): (voir Fig.3.1)

1ère étape: Description de la logique du circuit en termes d' opérateurs évolués tels que registres, compteurs, additionneurs, multiplexeurs, etc... A ce niveau, ces éléments sont indépendants de la technologie de réalisation. On parlera par exemple d'un "Compteur UP, n bits, synchrone, avec Load asynchrone".

2ème étape: On traduit cette description en un plan de câblage dans une technologie correspondant à des délais de réalisation les plus courts possibles. Aujourd'hui, ce sera une maquette à base de circuits MSI et SSI (dont le câblage et la mise au point peuvent prendre moins d'un mois). Ceci apporte une simulation logique en temps réel qui permet une validation très précoce, et permet de plus de travailler parallèlement sur les applications du circuit, ce qui en général rejaillit sur sa définition extérieure. Les allers-retours

notice-simulation sont rapides et court-circuitent toutes les étapes longues d'implantation.

3ème étape: On traduit la description de la première étape en un assemblage de briques dont le fonctionnement est prouvé et dont le dessin existe déjà. Il ne s'agit pas simplement de briques dont la forme est prévue pour un assemblage simple, mais elle doivent être conçues de telle manière que leur utilisation soit indépendante de leur structure interne. On doit pouvoir les utiliser comme un

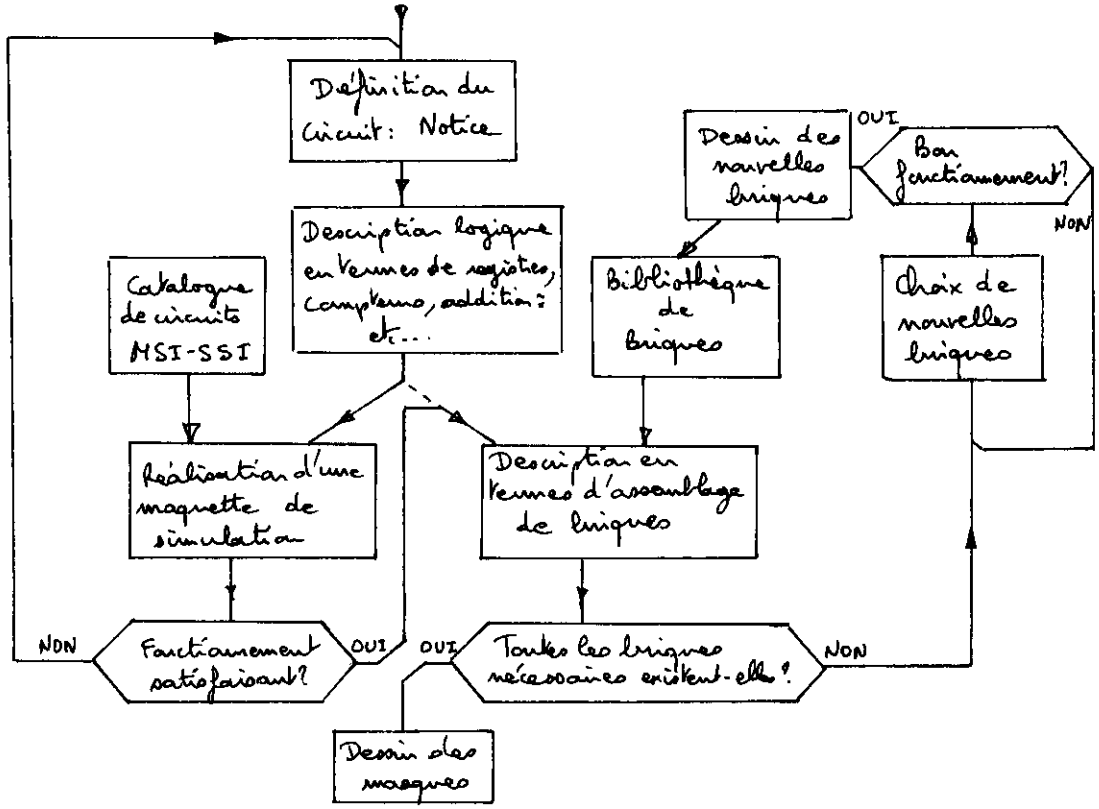


Fig 3.1

programmeur utilise des sous-programmes d'une bibliothèque standard. En particulier, ces sous-programmes ne doivent pas modifier des variables globales qui ne sont pas passées en paramètres. La fonction de ces briques doit pouvoir se définir par une relation booléenne liant les sorties aux entrées, modulo des considérations d'"entrance" et de "sortance" électriques. Ce n'est qu'à ce prix que la notion de brique simplifie la description des masques.

4ème étape: Le travail d'implantation du circuit se ramène alors à la définition des briques qui manquent à la bibliothèque, suivie de

leurs dessins et de leurs preuves.

Or, dans l'état actuel de la technologie, de telles briques, conçues pour éviter le plus possible au concepteur de considérer leur structure interne, sont presque obligatoirement en logique "statique". C'est à dire qu'elles n'utilisent pas d'astuces purement électroniques telles que: portes de transfert, condensateurs utilisés comme éléments de mémorisation, etc... qui ne réalisent pas la même fonction logique suivant leur environnement, et pour lesquelles les signaux d'entrées ne peuvent pas être quelconques, en particulier les horloges ont toujours une fréquence minimale de fonctionnement.

L'inconvénient de cette méthode est une perte de surface puisqu'alors on s'interdit des simplifications basées sur des raisonnement électroniques et on n'utilise pas pleinement les possibilités de la technologie. L'avantage réside dans une accélération importante de l'implantation puisqu'il suffit de connaître la faisabilité et les caractéristiques d'une porte élémentaire (le NOR dans la technologie utilisée). La simulation électrique n'est alors nécessaire que pour la mise au point de nouvelles briques ou l'utilisation aux limites des possibilités de vitesse ou de sortance électrique des briques standard.

Tous les circuits ne peuvent être traîtés suivant cette méthode. Une mémoire, pour laquelle la surface est très critique, mais où le temps de conception est celui de l'implantation de quelques dizaines de transistors vu le très grand taux de répétitivité, sera traité en logique dynamique, et on cherchera à utiliser le mieux possible les possibilités de la technologie. Par contre, un circuit dont la logique est disparate comme celle de ce circuit graphique et pour lequel la surface en logique statique est acceptable (économiquement) sera implanté beaucoup plus rapidement de cette façon. Le choix entre les

deux méthodes se fera par comparaison du coût du temps d'implantation au coût de fabrication des puces une fois le circuit implanté, le rendement de fabrication ρ étant lié à la surface A d'une puce par $\rho(A)=\rho_0 e^{-\sqrt{\frac{A}{A_0}}}$. Les arguments décisifs sont alors souvent plus économiques que techniques.

La suite de ce chapitre est l'établissement d'une bibliothèque standard de briques en logique statique, comprenant les fonctions les plus utilisées. Nous avons choisi d'y placer: un jeu de quelques bascules classiques, des éléments de compteurs, un additionneur, un comparateur. Les premiers paragraphes (3.3 à 3.8) étudient la structure logique de ces éléments indépendemment de leur position géométrique ou de leurs interconnexions, et donc sans les découper en briques. Le chapitre 3.9 de récapitulation recherche un découpage de ces fonctions. C'est là que les briques apparaissent, comme un compromis entre le travail du concepteur logique, qui par la suite ne manipulera plus qu'elles autant que cela sera possible, et le travail de l'implanteur en évitant que celui-ci n'implante plusieurs fois une même bascule, par exemple suivant qu'elle est utilisée seule ou dans un compteur.

3.3-CONTRAINTES CONDUISANT A LA STRUCTURE DES BRIQUES [8], [14]

Nous avons vu que la contrainte esentielle est la recherche de la minimisation de la surface du circuit. Une deuxième contrainte est la recherche de la vitesse, qui d'ailleurs se ramène souvent à une diminution de la surface puisque les temps de commutation sont proportionnels aux capacités électriques des éléments. Dans la technologie utilisée (MOS canal N, triple implantation), la réalisation des portes élémentaires est donnée figure 3.2. Les surfaces S_1, S_2, S_3 des transistors de types ①,②,③ sont reliées par: $S_2=4S_1$, et

$S_3 = 4nS_1$ si n est le nombre d'entrées de la porte NAND.

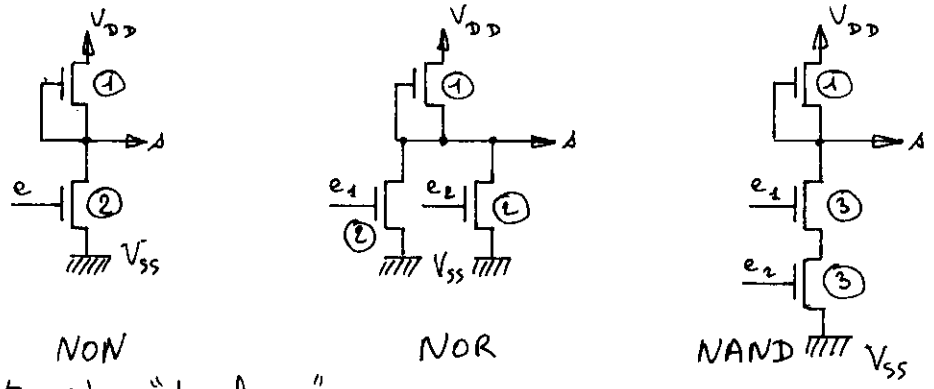


Fig 3.2

NON NOR NAND

① : transistors "de charge"
② et ③ : transistors "signal"

$V_{DD} > V_{SS}$

Il en découle immédiatement 2 conclusions:

-La surface d'un NOR varie linéairement en fonction du nombre de ses entrées. Sa vitesse est inversement proportionnelle à ce nombre, vu l'augmentation de capacité sur la sortie.

-Un NAND est plus gros qu'un NOR (et plus lent), et sa surface croît proportionnellement au carré du nombre de ses entrées.

On cherchera donc le plus possible à réaliser toutes les fonctions logiques à base de NORs ayant peu d'entrées.

Les fonctions de la forme $\overline{\sum_i (\prod_j e_{ij})}$ sont avantageuses car permettent un gain de transistors et un gain en vitesse. Par exemple, la fonction $s=\overline{e_1+(e_2\,e_3)}$ peut se réaliser comme sur la figure 3.3.

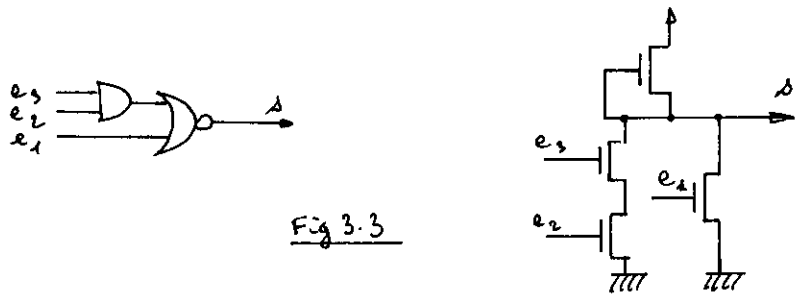


Fig 3.3

Mais, si on peut disposer des inverses des signaux d'entrées et si leur nombre est élevé, il vaut mieux n'utiliser que des NORs.

Note: Par la suite, nous représenterons les NORs allongés géométri-

quement comme sur la figure 3.4.a, chaque petit OU à une entrée représentant un transistor Signal, et le petit rond le transistor de Charge.

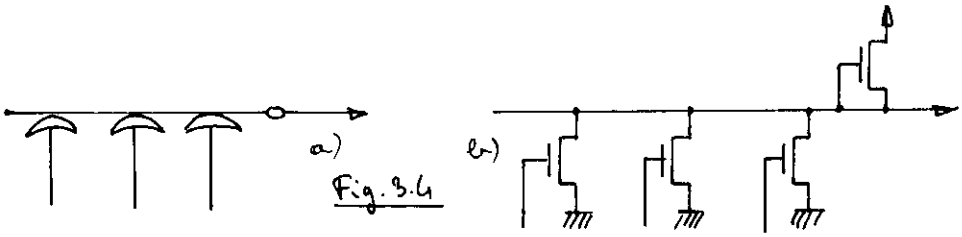


Fig. 3.4

Les structures logiques proposées ci-dessous sont partiellement orientées vers leur utilisation dans le circuit graphique. Certaines sont inspirées de celles de GASTINEL [8].

3.4-STRUCTURE DES BASCULES

a)Bascule R/S.

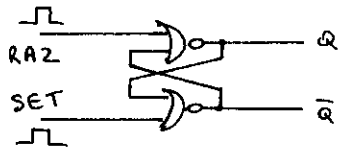


Fig 3.5

b)Latch.

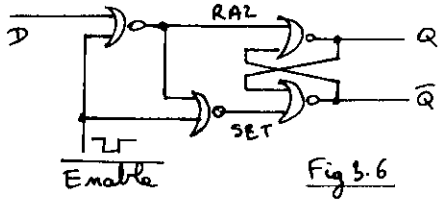


Fig 3.6

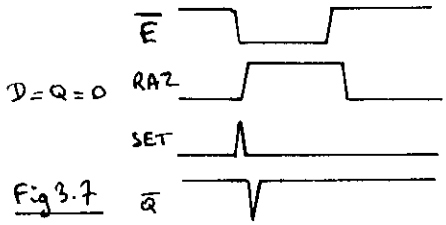


Fig 3.7

Le montage de la figure 3.6 (sans inverseur sur D) présente l'avantage de moins charger l'entrée D. Mais il peut générer des parasites sur ses sorties. Voir par exemple Fig.3.7 où D=Q=0.

c)Bascules non transparentes. 2 modèles de bascules non transparentes et de fonctionnement sûr sont utilisées: une est à entrées J-K, l'autre à entrée D.

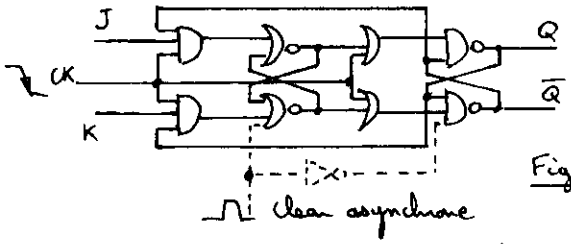


Fig 3.8

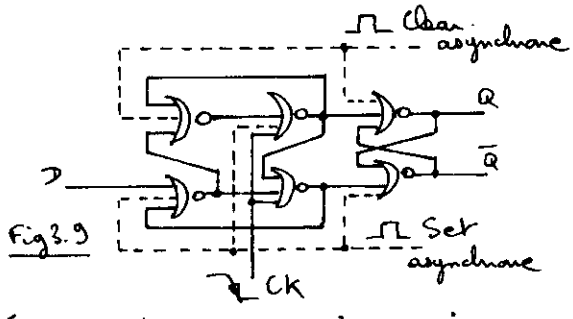


Fig 3.9

Ces 2 bascules sont également intéressantes, pour des raisons

différentes:

-La bascule J-K peut être plus petite que la bascule D car elle a un transistor de moins et beaucoup moins de connexions, mais alors elle est plus lente. A vitesse égale, la bascule D est plus petite car elle ne comporte que des NORs.

-La bascule D charge environ 8 fois moins l'entrée CK car celle-ci attaque 2 grilles de transistors du type ② au lieu de 4 du type ③ (Voir Fig.3.2).

-Le type d'entrée dicte éventuellement l'ajout de logique suplémentaire et est aussi un argument de choix.

d)Adaptation des entrées des bascules.

i)Entrée J-K sur une bascule D et vice-versa.

Fig 3.10

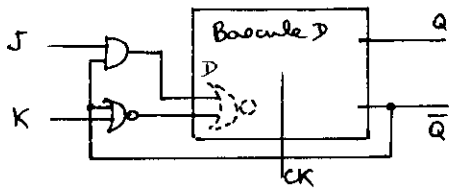
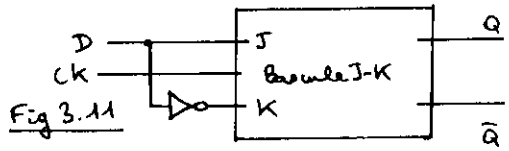
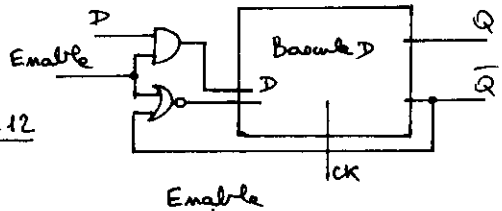


Fig 3.11



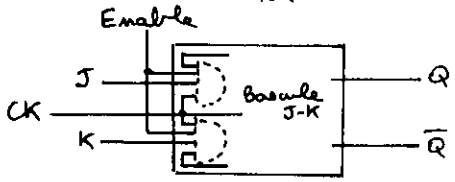
ii)Entrée Enable (ne retardant pas l'horloge).

Fig 3.12



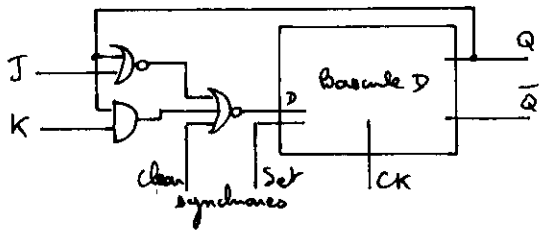
Si Enable est bas, la bascule se recopie

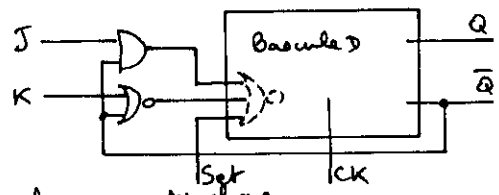
Fig 3.13

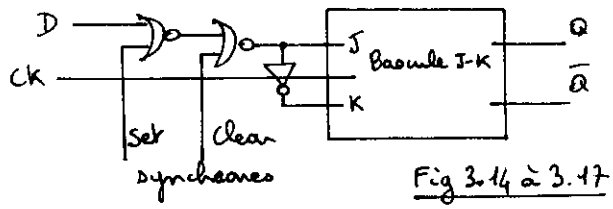


Dans ce cas, le Enable ne doit changer que si CK est bas.

iii)Bascules avec Clear ou Set synchrone et entrée D ou J-K.







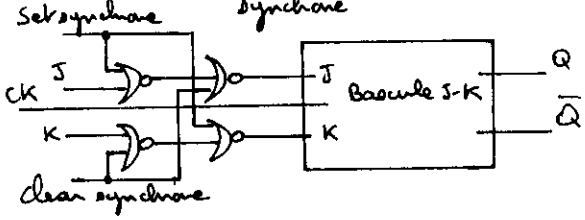


Fig 3.14 à 3.17

e)Conclusion. Nous avons choisi de n'utiliser que des bascules D pour 2 raisons. D'une part, la vitesse de fonctionnement du circuit est très élevée (1,75MHz), et ces bascules doivent être montées dans des compteurs 8 ou 12 bits! D'autre part, la configuration la plus utilisée pour ces compteurs est l'entrée T (entrées J-K réunies en une seule) avec Clear ou Set synchrone.

3.5-STRUCTURE DES COMPTEURS

Un compteur est constitué de bascules T. Sur l'entrée T d'un étage est appliqué le report des étages précédents. Deux méthodes sont utilisées pour propager ce report. On peut utiliser le ET des sorties de tous les étages précédents (ou le NOR de leurs sorties inverses):

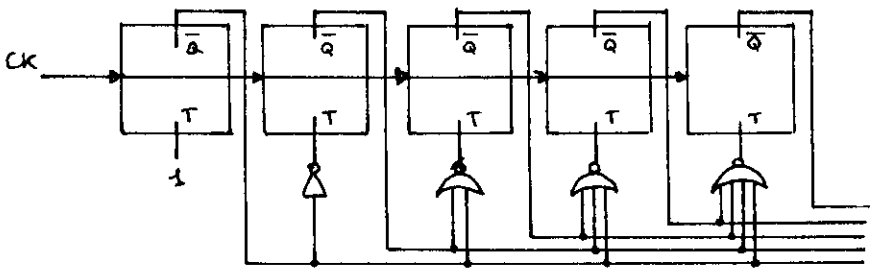


Fig 3.18

Chaque report est alors calculé en une seule couche logique. Mais cette couche n'est pas la même à tous les étages, et la répétitivité est supprimée. D'autre part, l'étage qui tourne le plus vite est le plus chargé capacitivement. De plus, pour un compteur long, la dimension des NORs des étages supérieurs nuit alors à la vitesse. Un compteur à propagation chaînée des reports peut être plus rapide. Il a l'avantage de s'implanter plus facilement.

a)Report de compteur UP. J. GASTINEL a utilisé 2 types de reports chaînés:

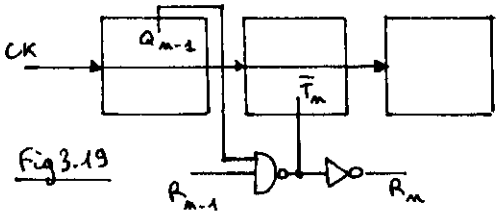


Fig 3.19

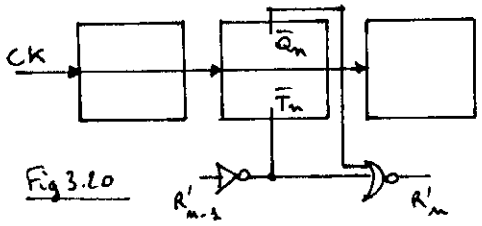


Fig 3.20

Deux couches logiques sont nécessaires par étage. Ce nombre peut être abaissé à un à condition de faire baisser le taux de répétitivité:

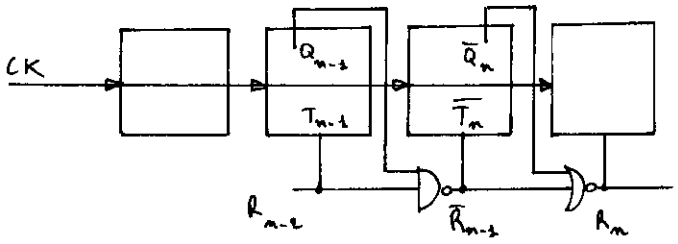


Fig 3.21

Les étages de rang pair et de rang impair sont "duaux". Mais alors, quitte à faire tomber la répétitivité, autant utiliser le schéma suivant qui calcule un report rapide sur 2 bits et le chaîne ensuite:

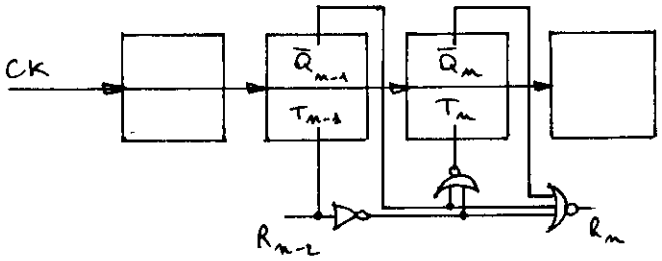


Fig 3.22

Ce schéma ne comporte que des NORs et qu'une seule couche logique par étage sur le report transmis. Toutes les bascules sont identiques.

b)<u>Report de compteur UP/DOWN.</u> Dans un compteur UP/DOWN, il faut doubler le calcul du report UP par un calcul de report DOWN. Si on utilise le montage de la figure 3.22, on peut en profiter pour faire remonter le taux de répétitivité, en décalant les 2 systèmes de calcul de reports:

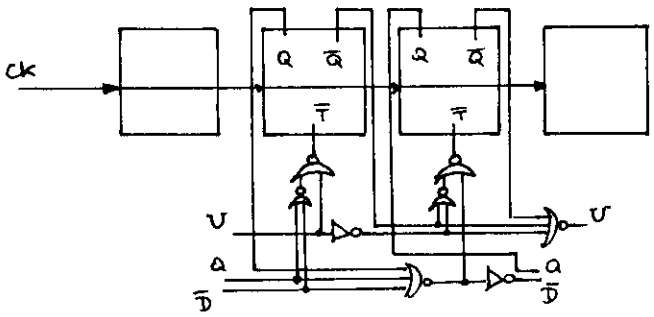


Fig 3.23

On obtiend alors un report pratiquement identique pour tous les étages (à une permutation près de Q et $\bar{Q}$):

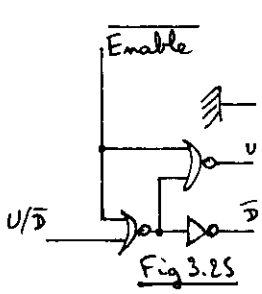


Fig 3.25

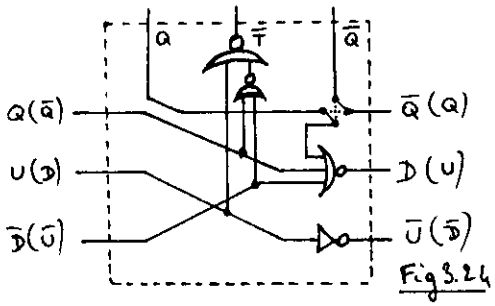


Fig 3.24

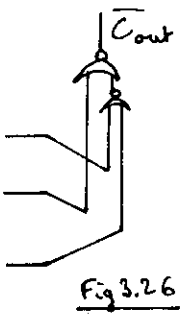


Fig 3.26

c)Reports d'entrée et de sortie. En entrée, il est du type dessiné en Fig.3.25, et dépend de la structure du premier étage. En sortie, le montage est analogue à celui du centre de la figure 3.24 (Fig.3.26). Il ne dépend pas de la structure du dernier étage.

3.6-CHOIX D'UNE STRUCTURE D'ADDITIONNEUR

a)Introduction. Le circuit graphique comprend 2 additionneurs 8 bits qui doivent calculer un report de sortie en 270ns (=T/2). Pour faciliter l'implantation, nous préférons utiliser un calcul de report chaîné qui a une structure plus répétitive. Mais pour assurer ce temps de calcul, il est préférable de chercher une structure à une couche logique par étage sur la transmission du report.

Nous avons utilisé un additionneur proposé par QUATSE et KEIR [15] en 1967, que nous décrivons ci-dessous après quelques rappels.

La somme S de 2 bits A et B se calcule à l'aide de 2 OUexclusifs

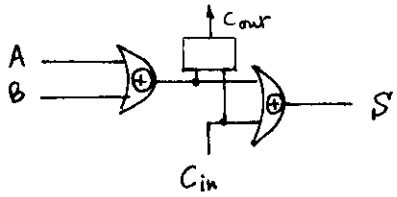


Fig 3.27

Le problème essentiel réside dans le calcul du report pour l'étage suivant (C_{out}). Généralement, on commence par calculer G et P définis par:

$$\begin{cases} G=A.B & \text{(G est mis pour "générateur", et} \\ \bar{P}=\overline{A+B} & \text{P pour "propagateur".)} \end{cases}$$

qui ont l'avantage de permettre facilement le calcul de ce report, mais aussi du OUexclusif de A et B. En effet:

$$\begin{cases} A\oplus B=\overline{G+\bar{P}} & (\text{car } A\oplus B=\bar{A}B+A\bar{B}=(\bar{A}+\bar{B}).(A+B)=\overline{AB}.(A+B)=\overline{AB+(\overline{A+B})}) \\ C_{out}=G+PC_{in} & \text{(ce qui justifie les noms de G et P)} \end{cases}$$

Pour une logique à NORs, on utilise indifféremment les 2

formules suivantes (exemple Fig.3.28) suivant la polarité souhaitée

$$\begin{cases} \overline{C_{out}} = \overline{G + \overline{\overline{P} + \overline{C}}}_{in} \\ C_{out} = \overline{\overline{P} + \overline{\overline{G} + \overline{C}}}_{in} \end{cases} \quad (\text{car } C_{out} = G + PC_{in} = GP + PC_{in} = P(G + C_{in}))$$

pour les reports.

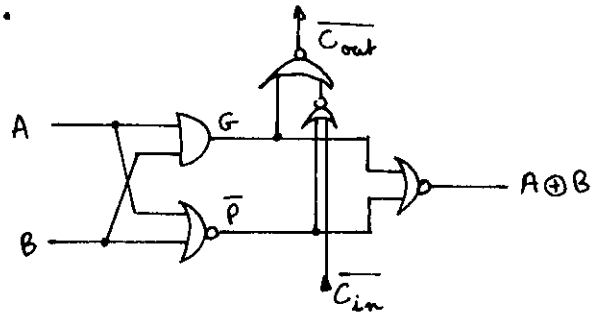


Fig 3.28

b)Propagation du report en une couche logique. L'inconvénient de ce montage est l'existence de 2 couches logiques pour transmettre le report. QUATSE et KEIR ont proposé de transmettre l'information de report à l'aide de plusieurs variables.

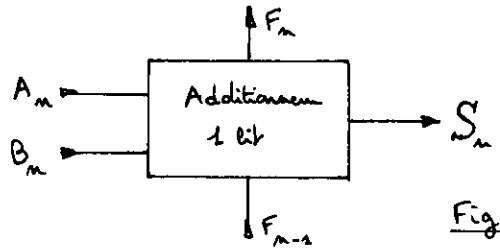
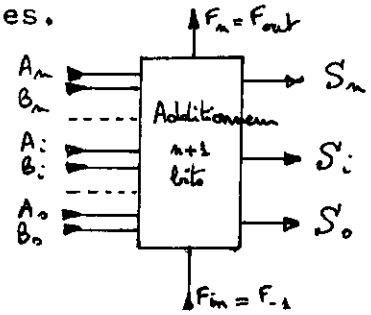


Fig 3.29



En utilisant les notations précisées par la figure 3.29, reprenons les formules de transmissionde report sous la forme:(en changeant la polarité du report entre 2 étages consécutifs)

$$\begin{cases} C_{n+2} = G_{n+2} + \overline{\overline{P_{n+2}} + \overline{C_{n+1}}} \\ \overline{C_{n+1}} = \overline{P_{n+1}} + \overline{G_{n+1} + C_n} \end{cases} \quad (1)$$

Cherchons 2 fonctions α_n et $\overline{\beta_n}$ telles que

$$\begin{cases} C_n = G_n + \alpha_n \quad (= \overline{\overline{G_n} \cdot \overline{\alpha_n}}) \\ \overline{C_n} = \overline{P_n} + \overline{\beta_n} \quad (= \overline{P_n \cdot \beta_n}) \end{cases} \quad (2)$$

Reportons (2) dans (1). Si α_n et $\overline{\beta_n}$ existent, elles vérifient:

$$\begin{cases} G_{n+2} + \alpha_{n+2} = G_{n+2} + \overline{\overline{P_{n+2}} + \overline{P_{n+1}} + \overline{\beta_{n+1}}} \\ \overline{P_{n+1}} + \overline{\beta_{n+1}} = \overline{P_{n+1}} + \overline{G_{n+1} + G_n + \alpha_n} \end{cases} \quad (3)$$

Une solution est: $\begin{cases} \alpha_m = \overline{\overline{P_m} + \overline{P_{m-1}} + \overline{\beta_{m-1}}} \\ \overline{\beta_m} = \overline{G_m + G_{m-1} + \alpha_{m-1}} \\ \alpha_{-1} = G_{-1} = P_{-1} = \beta_{-1} = C_{in} \end{cases}$

Ceci permet de réaliser un report en une couche logique, transmis par 2 variables:

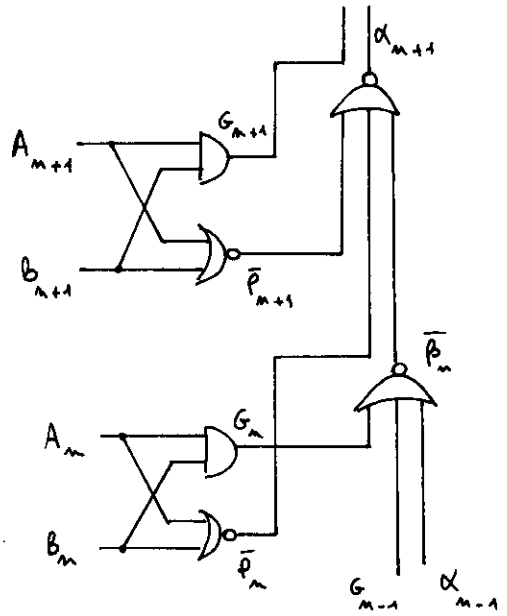


Fig 3.30

Ceci est par ailleurs un bon montage de comparateur (voir §3.7)

Pour calculer la somme S, il suffit de retrouver le "carry" habituel par les formules (2):

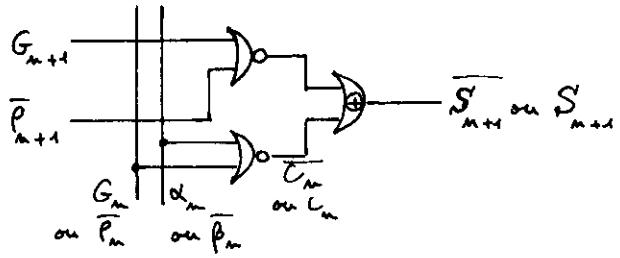


Fig 3.31

Mais QUATSE et KEIR ont cherché d'autres fonctions de report α'_m et $\overline{\beta'_m}$ qui vérifient aussi:

$$(2') \quad \begin{cases} C_m = G_m + \alpha_m = G_m + \alpha'_m \Rightarrow \overline{G_m} \cdot \overline{\alpha'_m} = \overline{G_m} \cdot \overline{\alpha_m} \\ \overline{C_m} = \overline{P_m} + \overline{\beta_m} = \overline{P_m} + \overline{\beta'_m} \end{cases}$$

etqui doivent donc vérifier (3). Une solution est:

$$(4') \quad \begin{cases} \alpha'_m = \overline{\overline{P_m} + G_m + \overline{P_{m-1}} + \overline{\beta'_{m-1}}} \\ \overline{\beta'_m} = \overline{\overline{P_m} + G_m + G_{m-1} + \alpha'_{m-1}} \\ \alpha'_{-1} = G_{-1} = P_{-1} = \beta'_{-1} = C_{in} \end{cases}$$

Or, α'_m et $\overline{\beta'_m}$ ont l'avantage de permettre un calcul plus simple de S. En effet: (et ce calcul peut se faire de 2 façons différentes)

$$\overline{\alpha'_m} = (\overline{P_m} + G_m) + (\overline{P_{m-1}} + \overline{\beta'_{m-1}})$$
$$= \overline{A_m \oplus B_m} + \overline{C_{m-1}}$$

et de même, $\beta'_m = \overline{A_m \oplus B_m} + C_{m-1}$

Or, $S_m = (A_m \oplus B_m) \oplus C_{m-1}$

$= \left[(A_m \oplus B_m) \cdot \overline{C_{m-1}}\right] + \left[\overline{(A_m \oplus B_m)} \cdot C_{m-1}\right]$

$= \left[(A_m \oplus B_m) \cdot \alpha'_m\right] + \left[\alpha'_m \cdot C_{m-1}\right]$

$= \overline{A_m \oplus B_m} + \alpha'_m + \overline{\alpha'_m + C_{m-1}}$

$S_m = \overline{G_m + \overline{P_m} + \alpha'_m} + \overline{\alpha'_m + \overline{P_{m-1}} + \beta'_{m-1}}$

$= \overline{(A_m \oplus B_m) \oplus C_{m-1}}$

$= \overline{\left[(A_m \oplus B_m) \cdot C_{m-1}\right] + \left[\overline{(A_m \oplus B_m)} \cdot \overline{C_{m-1}}\right]}$

$= \overline{((A_m \oplus B_m) \cdot \beta'_m) + (\beta'_m \cdot \overline{C_{m-1}})}$

$= \overline{A_m \oplus B_m} + \overline{\beta'_m} + \overline{\beta'_m + C_{m-1}}$

$= \overline{G_m + \overline{P_m} + \beta'_m} + \overline{\beta'_m + G_{m-1} + \alpha'_{m-1}} = S_m$

Dans ce cas, le "carry" n'est jamais calculé, A⊕B non plus, et la somme s'obtiend en 3 portes:

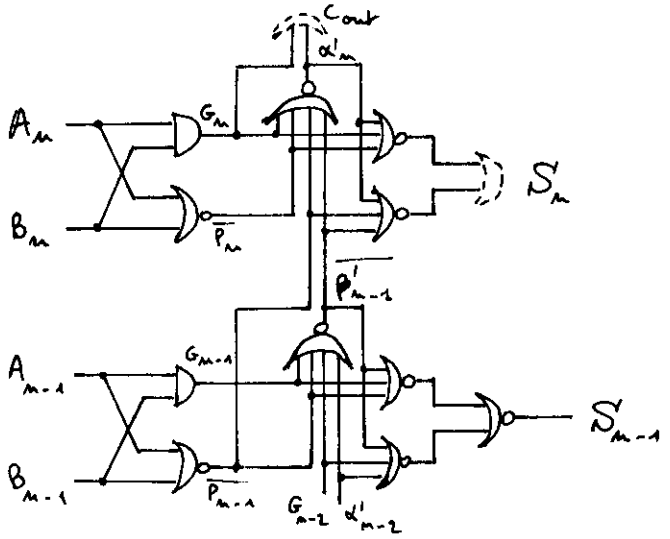


Fig 3.32

Pour un additionneur à 8 étages, il faut:

-9 couches logiques sur le report de sortie,

-10 couches logiques sur S_8.

c)<u>Identité des étages pairs et impairs.</u> Le seul inconvénient de ce dernier additionneur est la différence de structure entre les étages pairs et impairs. Mais on peut s'en affranchir en transmettant à la fois α'_m et $\overline{\beta'_m}$ pour tous les étages.

En effet: $S_m = C_{m-1} \oplus (A_m \oplus B_m)$

$= (C_{m-1} \cdot \overline{A_m \oplus B_m}) + (\overline{C_{m-1}} \cdot A_m \oplus B_m)$

$= (C_{m-1} \cdot \overline{\alpha'_m}) + \overline{\beta'_m}$

$$= \overline{\overline{C_{m-1} + \alpha'_m} + \overline{\beta'_m}}$$

$$S_m = \overline{\overline{P_{m-1} + \overline{\beta'_{m-1}} + \alpha'_m} + \overline{\beta'_m}}$$

(de même, on peut montrer , ou déduire par dualité que

$$\overline{S_m} = \overline{G_{m-1} + \alpha'_{m-1} + \overline{\overline{\beta'_m} + \alpha'_m}}$$

ce qui donne le montage de la figure 3.33

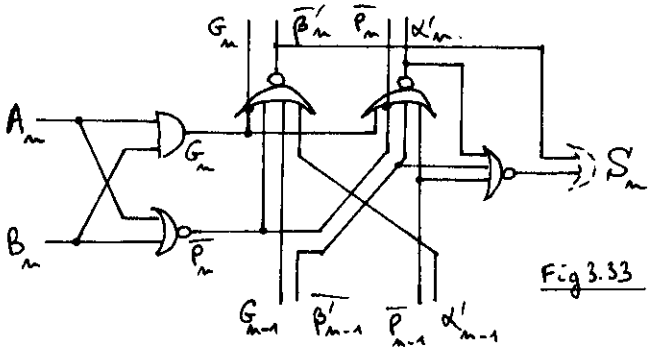


Fig 3.33

On obtiend un montage plus petit et plus rapide car les sorties α'_m, $\overline{\beta'_m}$, G_m et $\overline{P_m}$ chargent moins d'entrées.

Le ET servant à calculer G_m peut être remplacé par 2 NORs, à condition de disposer d'une entrée inversée et une seule:

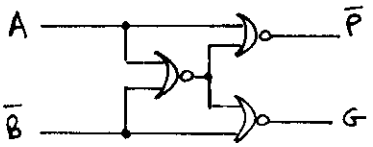


Fig 3.34

En effet:

$$\overline{P} = \overline{A}.\overline{B} = \overline{A}.(A+\overline{B}) = \overline{\overline{\overline{A+\overline{A}+\overline{B}}}}$$

$$G = AB = B.(A+\overline{B}) = \overline{\overline{B}+\overline{\overline{A}+\overline{B}}}$$

La permutation de A et B entraîne une permutation de G et $\overline{P}$.

3.7-COMPARATEURS

L'égalité entre 2 mots de n bits A et B est:

$$E = \prod_i (A_i = B_i)$$

$$= \prod_i (\overline{A_i \oplus B_i})$$

$$= \prod_i (G_i + \overline{P_i})$$ où G et $\overline{P}$ sont définis de la même façon que dans le paragraphe précédent. Ce qui donne:

$$E = \overline{\sum_i (\overline{G_i + \overline{P_i}})}$$ pour une logique à NORs (Fig.3.35)

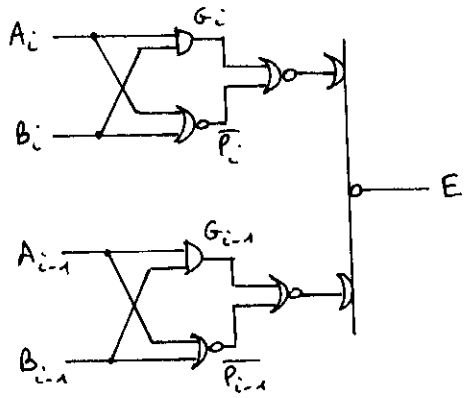


Fig 3.35

Si l'on dispose de A et $\overline{B}$ (complément à 1 du mot B), on peut utiliser le montage de la figure 3.34, ou mieux celui de la figure 3.36, en considérant que:

$$E = \prod_i (\overline{A_i \oplus B_i})$$
$$= \prod_i (A_i \oplus \overline{B_i})$$
$$= \overline{\sum_i (A_i \oplus \overline{B_i})} = \overline{\sum_i G'_i + \overline{P'_i}} \qquad \text{où } G'_i = A_i \overline{B_i} \text{ et } \overline{P'_i} = \overline{A_i + \overline{B_i}}$$
$$= \overline{\sum_i A_i \overline{B_i} + \overline{A_i + \overline{B_i}}}$$

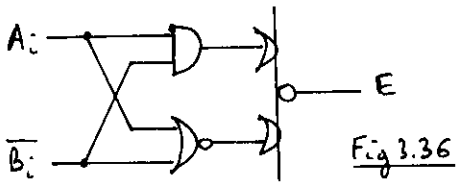


Fig 3.36

Ce schéma utilise 6 transistors par étage (au lieu de 8 pour la Fig.3.35).

Un comparateur arithmétique se réalise à partir d'un additionneur. En effet, le report de sortie de l'additionneur de la figure 3.37 est:

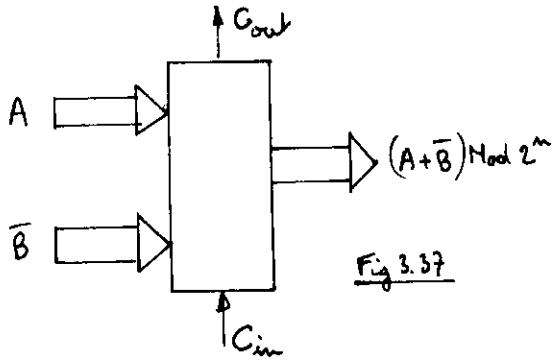


Fig 3.37

n est le nombre de bits de l'additionneur.

$$C_{out} = 1 \text{ si } A + \overline{B} + C_{in} \geqslant 2^n$$

or, $\overline{B} = 2^n - B - 1 \quad \Longrightarrow \quad A + 2^n - B - 1 + C_{in} \geqslant 2^n$

$$\Longrightarrow \quad A - B + C_{in} > 0$$

Donc, si C_{in}=0 alors C_{out} représente A > B et

si C_{in}=1 alors C_{out} représente A ⩾ B.

On peut utiliser le report de la figure 3.30 à une porte par étage.

3.8-MEMOIRE MORTE

Une mémoire morte à mots de 1 bit peut se réaliser comme indiqué figure 3.38. La matrice est constituée de longs NORs horizontaux attaqués par des grilles verticales. Chaque bit est codé par l'abscence ou la présence d'un transistor.

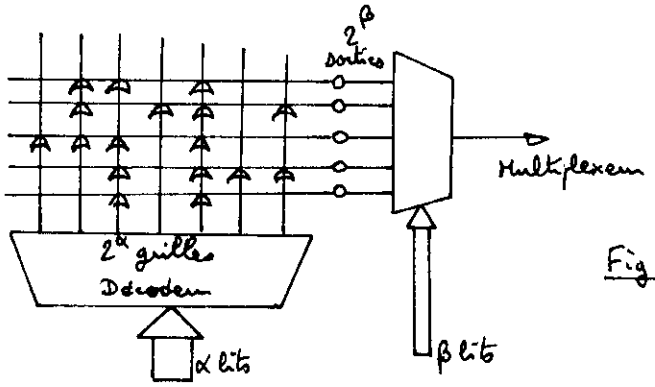


Fig 3.38

Le schéma à NORs détaillé de la mémoire est:

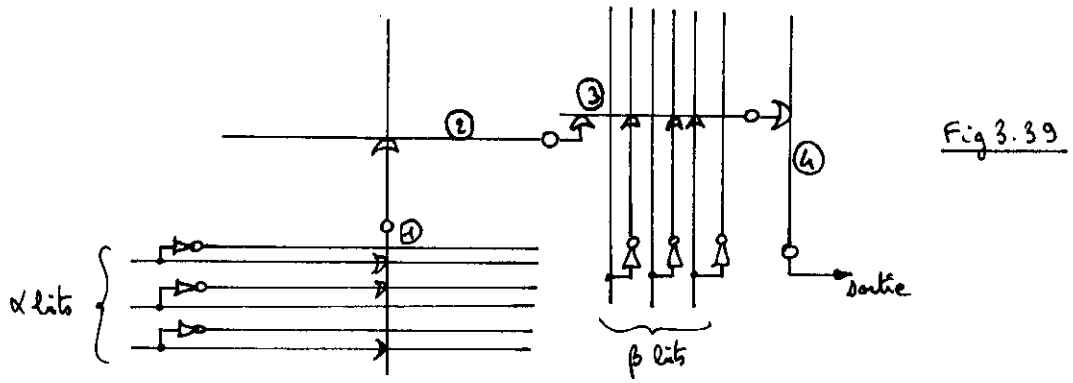


Fig 3.39

L'ensemble comprend 4 couches logiques: ①, ②, ③, ④. La couche ① est lente car les NORs attaquent de longues grilles très chargées capacitivement. La couche ② est lente car la sortie des NORs est aussi très chargée. La couche ③ a la vitesse d'un NOR à β+1 entrées. La couche ④ a la vitesse d'un NOR à 2^{β} entrées.

3.9-RECAPITULATION: BIBLIOTHEQUE DE BRIQUES

Ce paragraphe est la liste des briques retenues après les choix faits dans les paragraphes précédents. Elles sont choisies pour permettre d'en manipuler le minimum lors de la conception logique, tout en minimisant le travail d'implantation. Par exemple, les variantes d'une même bascule s'obtiennent en assemblant à un même "noyau de bascule" des briques "entrée de bascules" différentes.

La définition de ces briques suppose lors de l'implantation le respect des règles suivantes:

-Toutes les briques qui suivent (nº1 à 13) devront après implantation avoir la même hauteur (environ 150μ), la largeur dépendant de la complexité.

-La position relative des fils de communication avec les briques voisines devra être respectée, pour que chacune puisse se connecter à un maximum de briques.

-Les connexions traversant une brique sans contact interne à celle-ci devront être implantées en vue de la connexion entre elles de futures briques latérales à la brique considérée.

a) Bascule R/S.

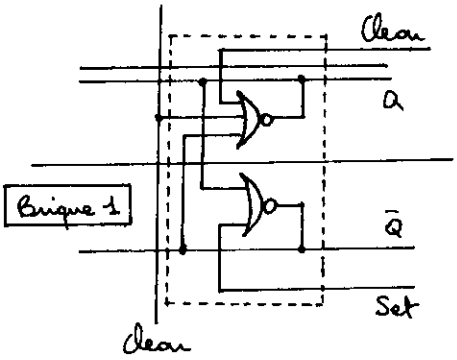


Fig 3.40

que l'on peut aussi dessiner:

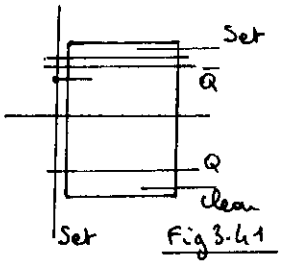


Fig 3.41

b) Load asynchrone et sortie 3 états.

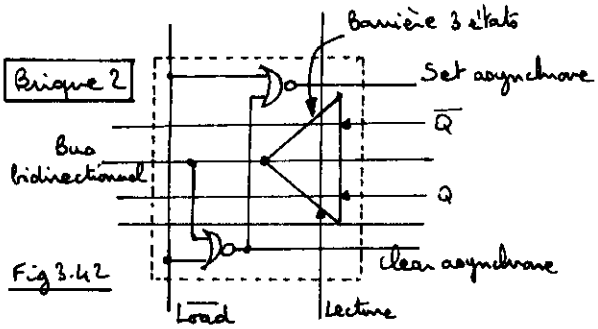
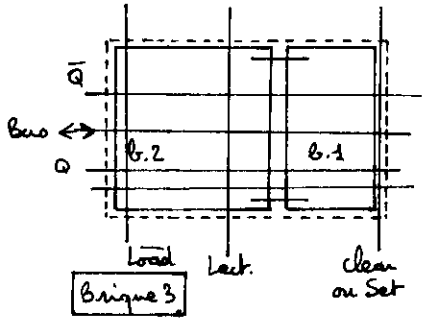


Fig 3.42

Cette disposition permet le couplage au bus 3 états interne de toute bascule ayant un Clear et un Set asynchrone.

c) Latch à sortie 3 états et Clear ou Set.



3.1 : avec Clear
3.2 : avec Set

Fig 3.43

Cette brique s'obtiend à partir des briques 1 et 2 (avec renversement de la brique 1 si Set).

d) Bascule D-D'.

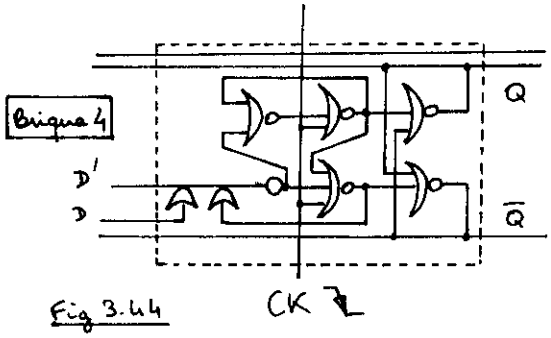


Fig 3.44

D' est la sortie du NOR et permet de rajouter des entrées. voir Fig.3.2 et 3.4.

e)Bascule D avec Clear et Set asynchrones.

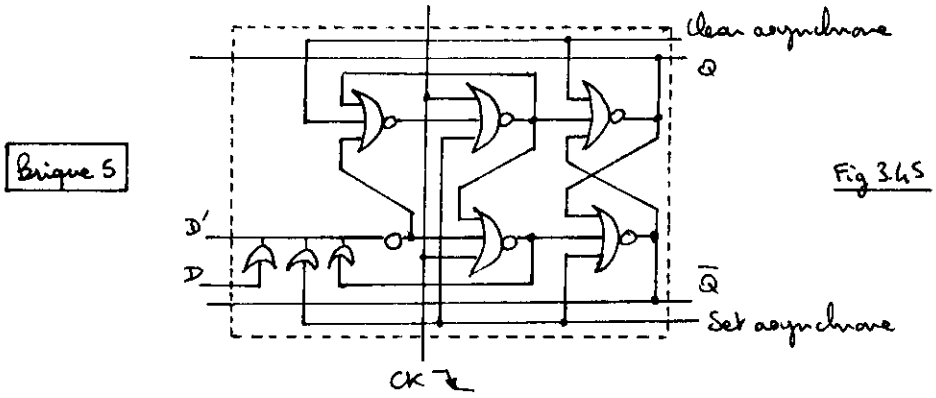


Fig 3.45

f)Bascule à Set sur front et Clear asynchrone.

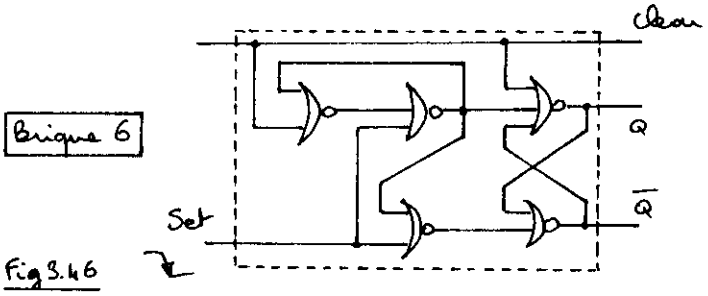


Fig 3.46

g)Bascule D avec Enable.

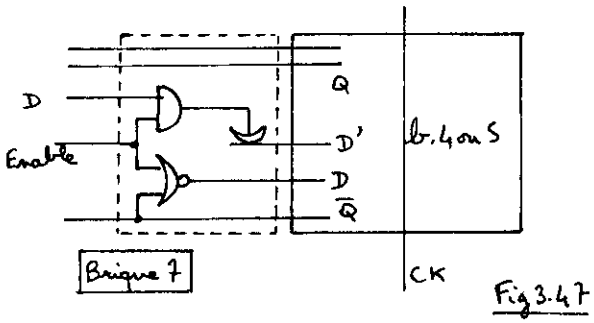


Fig 3.47

voir Fig 3.12

h)Bascule T ou $\overline{T}$ avec Clear ou/et Set synchrone.

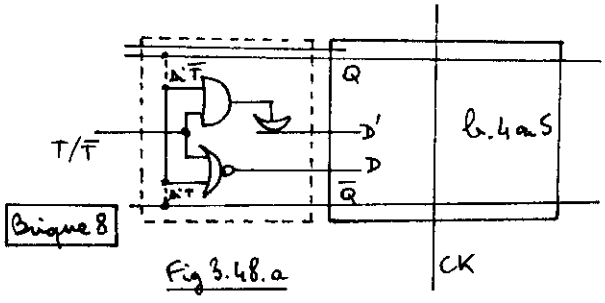


Fig 3.48.a

8.1: T sans Clear ni set
8.2: $\overline{T}$ sans Clear ni set

voir Fig 3.14 à 3.17

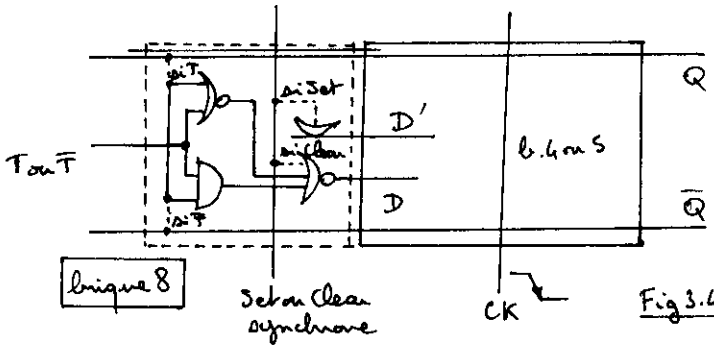


Fig 3.48.b.

8.3 : T avec Clear
8.4 : T avec Set
8.5 : F̄ avec Clear
8.6 : F̄ avec Set
8.7 : T avec Clear et Reset

i) Bascule T ou $\overline{T}$ avec Load synchrone.

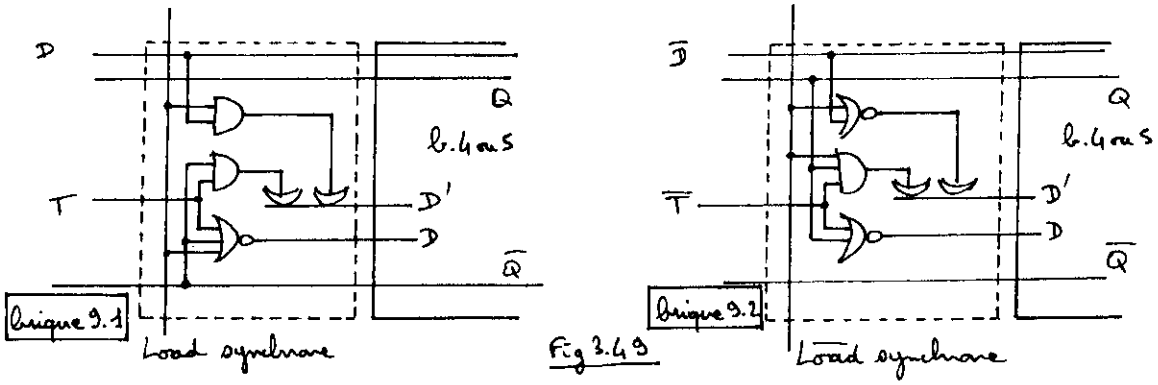


Fig 3.49

Quand Load est haut, T doit être bas.

j) Report de compteur **UP.** 4 briques sont nécessaires pour couvrir toutes les possibilités de disposition des bascules.

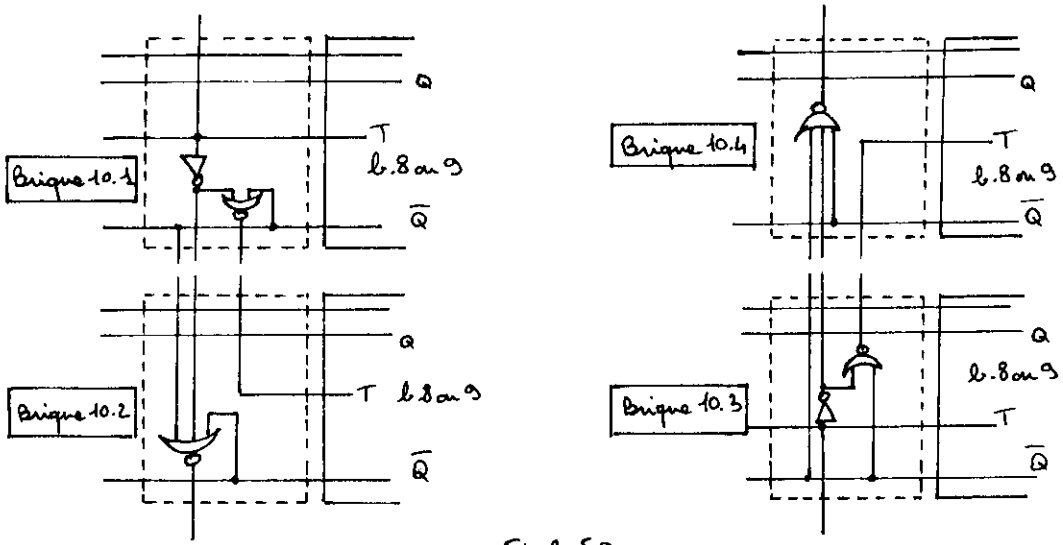


Fig 3.50

k)Report de compteur UP/DOWN. Ces briques se déduisent d'un même dessin (Fig.3.24) modulo 2 contacts et des symétries. Il faut leur assortir des reports comme indiqué sur les figures 3.25 et 3.26.

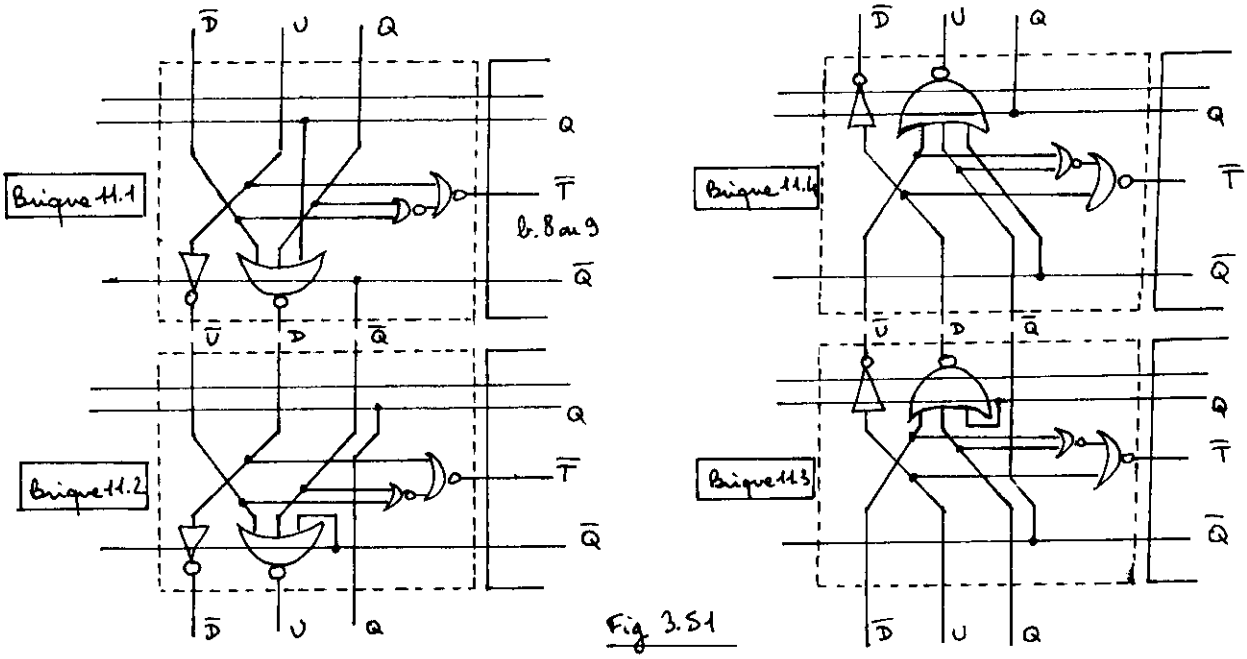
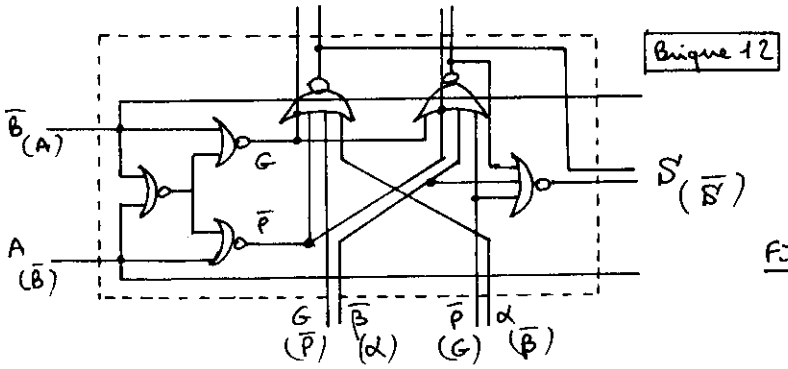


Fig 3.51

l)Additionneur.



$C = G + \alpha$

$\bar{C} = \bar{P} + \bar{\beta}$

Fig 3.52

En intervertissant A et $\bar{B}$, on intervertit G et $\bar{P}$. Si l'on change aussi les fonctions de report (valeurs entre parenthèses), alors on obtiend $\bar{S}$ d'après la remarque du paragraphe 3.6.c.

m)Comparateur d'égalité.

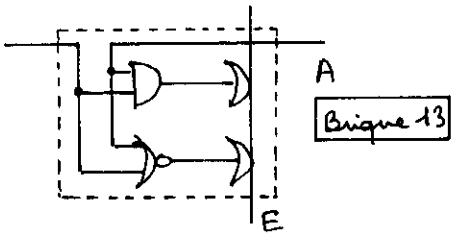


Fig 3.53

3.10-PLAN DE CHACUN DES CHAPITRES SUIVANTS

Les chapitres qui suivent décrivent le fonctionnement précis de chacun des automates du circuit: partie visualisation, générateur de vecteurs, générateur de caractères, circuits de contrôle. Pour chaque chapitre, nous donnons une description indépendante de la réalisation, un schéma de maquette MSI-SSI, la structure de briques nouvelles nécessaires à l'implantation, puis la disposition des briques. Le dernier chapitre contient en outre la disposition de ces blocs sur la puce.

4-CONTROLEUR DE VISUALISATION-SYNCHRONISATION ET GESTION DE LA MEMOIRE D'IMAGE

4.1-PRESENTATION

Ce contrôleur réalise le bloc de la figure 2.19. Il comprend essentiellement:

-Un compteur synchrone qui génère les adresses de visualisation, les signaux de synchronisation TV, et les signaux de synchronisation du contrôleur d'écriture.

-Une logique combinatoire d'adressage de la mémoire soit à partir du compteur de visualisation, soit à partir de l'adresse d'écriture, sur les 2 groupes de sorties IAD et ISL.

Le compteur de visualisation-synchronisation orchestre toutes les opérations réalisées par le circuit intégré. Il est modifié sur le front descendant de CKIN (dont la période sera appellée T). Nous pouvons le découper conceptuellement en une partie basse dont la période est une ligne TV (=H), qui sera appellé compteur horizontal; et une partie haute dont la période est une image TV (=625H) qui sera appellé compteur vertical.

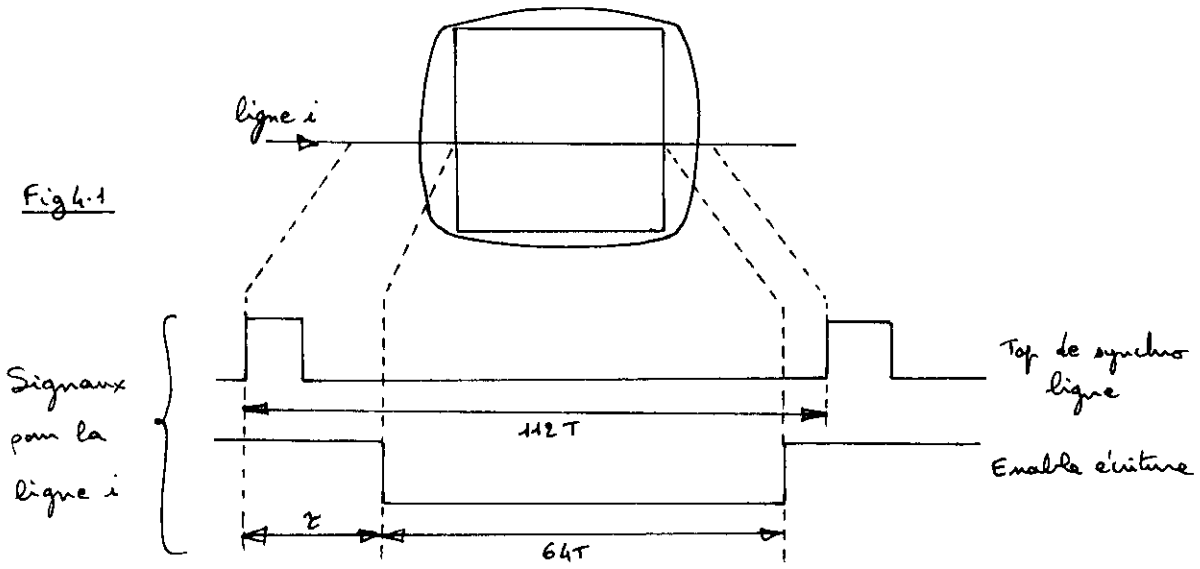
A-COMPTEUR DE VISUALISATION-SYNCHRONISATION

4.2-COMPTEUR HORIZONTAL

Une ligne est constituée de $112_{(10)}$ T (dans ce chapitre, les entiers sont quelquefois représentés avec un indice indiquant la base de numération), dont 64T de visualisation et 48T d'écriture. Cette partie du compteur peut donc se réaliser avec 7 étages

(puisque $2^6 < 112 < 2^7$). Appellons s_0-s_6 les sorties de ce compteur.

Fig 4.1



Il sera réalisé par un compteur binaire classique (UP) sur lequel la reconnaissance de la valeur $111_{(10)}$ entraînera une remise à zéro synchrone (Clear). Comme les 6 sorties de poids faibles doivent constituer l'adresse mémoire horizontale, ce Clear doit se faire sur le bord gauche de l'image affichée. La sortie s_6 sera donc exactement le signal "Enable écriture" ($\overline{\text{IALL}}$) puisque la visualisation comprend $2^6 = 64T$).

a)<u>Signaux à générer.</u> Compte tenu de cette dernière contrainte, une fois que nous avons choisi le cadrage de l'image (déphasage τ), les valeurs du compteur à reconnaître pour la synchronisation nous sont imposées (Fig.4.2). Ces valeurs sont de 2 types: d'une part les reports pour le compteur vertical, d'autre part les "tops" de synchronisation ligne ainsi que les tops d'égalisation et de serration apparaissant lors du top trame.

En symbolisant chaque reconnaisseur par un grand ET pour simplifier, le schéma de ce compteur est donné Fig.4.3.

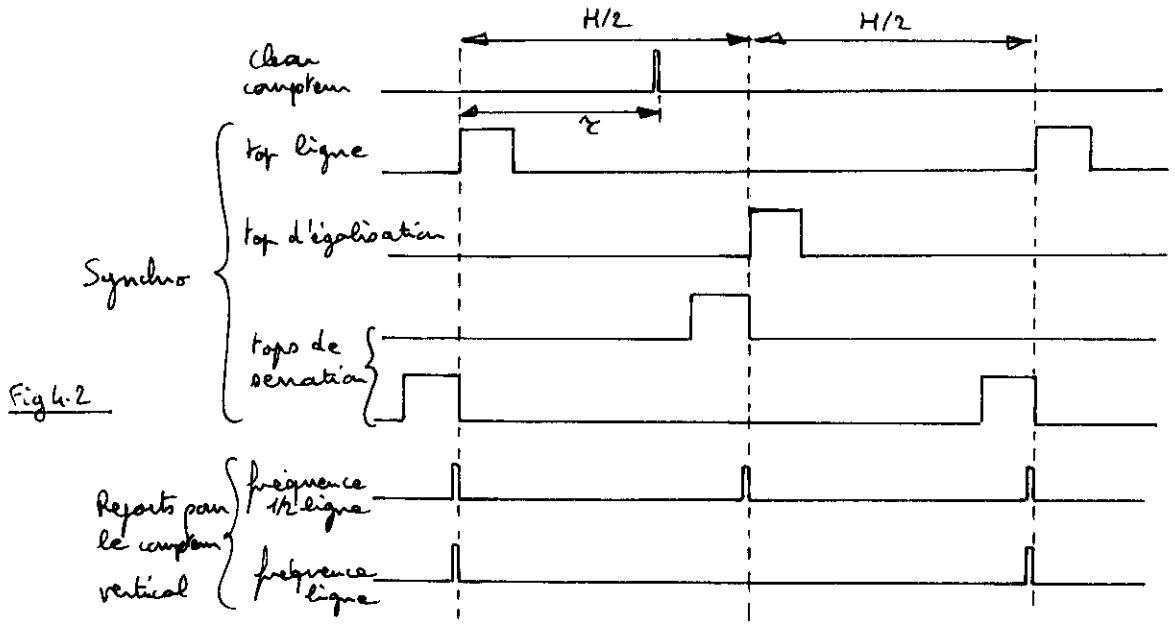


Fig 4.2

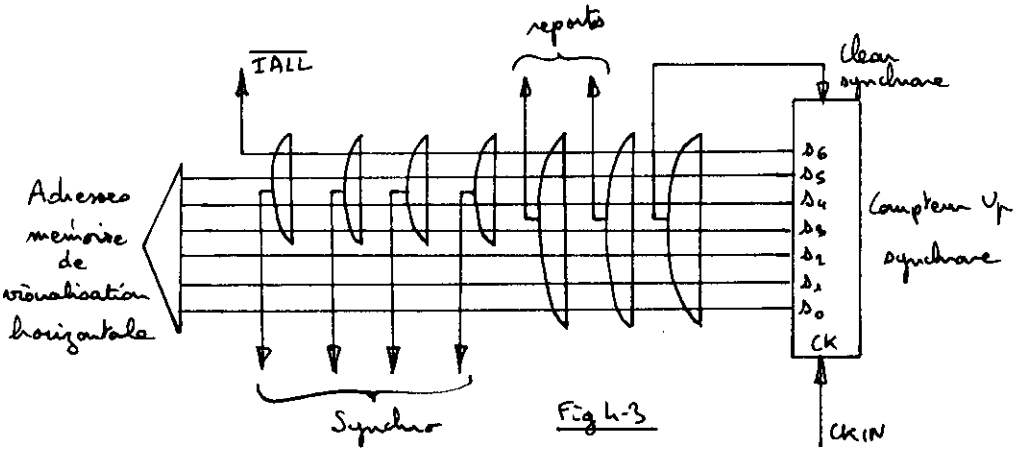


Fig 4-3

b)Utilisation de bascules de retards. Ces reconnaisseurs peuvent se simplifier, du point de vue l'implantation sur le circuit intégré, par la remarque de GASTINEL [8] qui a utilisé des "bascules de retard". Il suffit alors de 4 reconnaiseurs plus petits que les précédents et de 2 bascules de retard qui décalent les signaux par sauts de 8 périodes d'horloge. Ce sont des bascules D avec Enable, celui-ci étant le ET des 3 bits de poids faibles du compteur (figures 4.4 et 4.5).

Les signaux à générer s'obtiennent de la façon suivante:

-Clear: $\beta . \alpha$

-top ligne: γ décalé une fois ($=\gamma'$)

-top d'égalisation: δ décalé une fois ($=\delta'$)

-top de serration: $\gamma + \delta$

-report ligne: $\gamma \cdot \alpha$

-report demi-ligne: $(\gamma+\delta) \cdot \alpha$

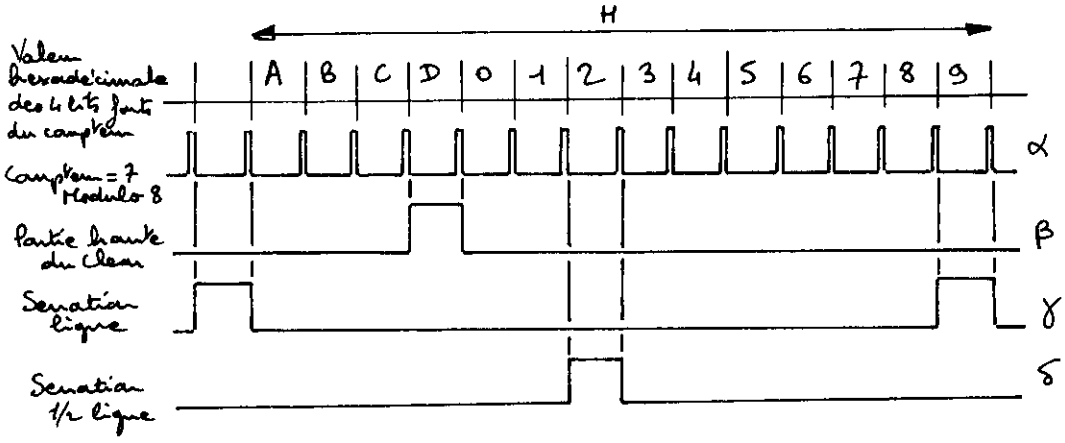


Fig 4-4

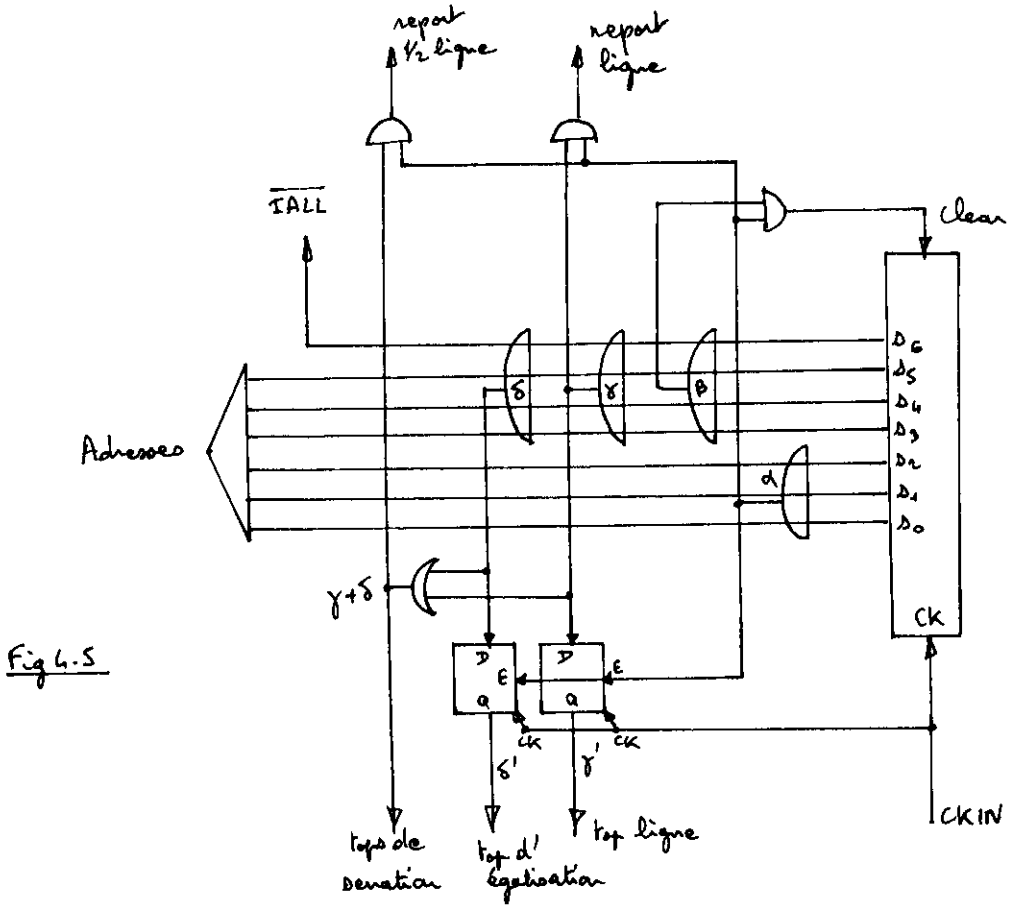


Fig 4-5

4.3-COMPTEUR VERTICAL

La période du compteur vertical est 625H. L'évènement le plus fin à reconnaître est de durée H/2, puisque la moitié des tops trame commencent au milieu d'une ligne. Le nombre d'états de ce compteur est donc 1250 et il comprend 11 bits puisque $2^{10} < 1250 < 2^{11}$.

a)Contraintes de séquencement. Il faut régler le séquencement de ces 1250 états dans les buts suivants:

-9 sorties de ce compteur doivent pouvoir fournir directement les adresses de visualisation verticales pour la mémoire d'image. Le bit de poids fort indique la parité des lignes dans le cas de trames entrelacées. Il correspond au bit faible de l'adresse verticale dans le format 512x512. Il est ignoré dans le cas de trames non entrelacées (format 256x256 et inférieurs).

-Le top trame doit être facile à générer.

-Un bit doit indiquer les bords haut et bas de l'image pour participer au signal IFON et pour générer le signal "fin trame".

La réalisation qui vient le plus rapidement à l'esprit est de prendre un simple compteur 11 bits incrémenté toutes les demi-lignes.

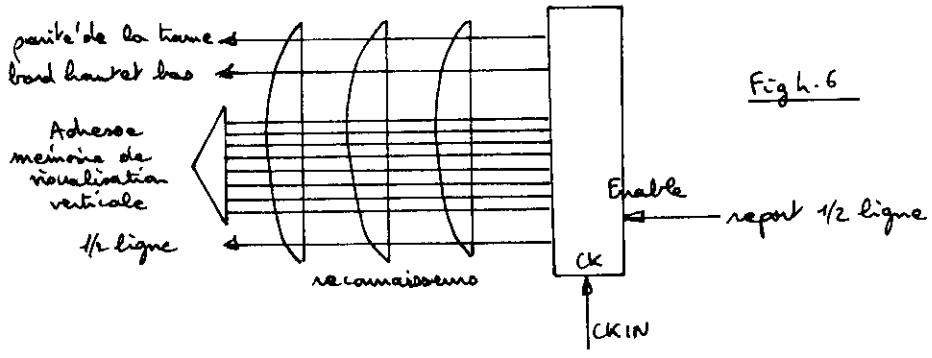


Fig.6

Ce système a le grave inconvénient suivant: comme une trame contient un nombre impair de demi-lignes (625), il découle que lors d'une

trame sur deux, les sorties de ce compteur changent au milieu de la ligne et sont donc inutilisables pour l'adressage vertical des mémoires.

Les 10 bits de poids fort de ce compteur doivent donc être incrémentés en fin de ligne, alors que seul le bit faible doit basculer à chaque demi-ligne.

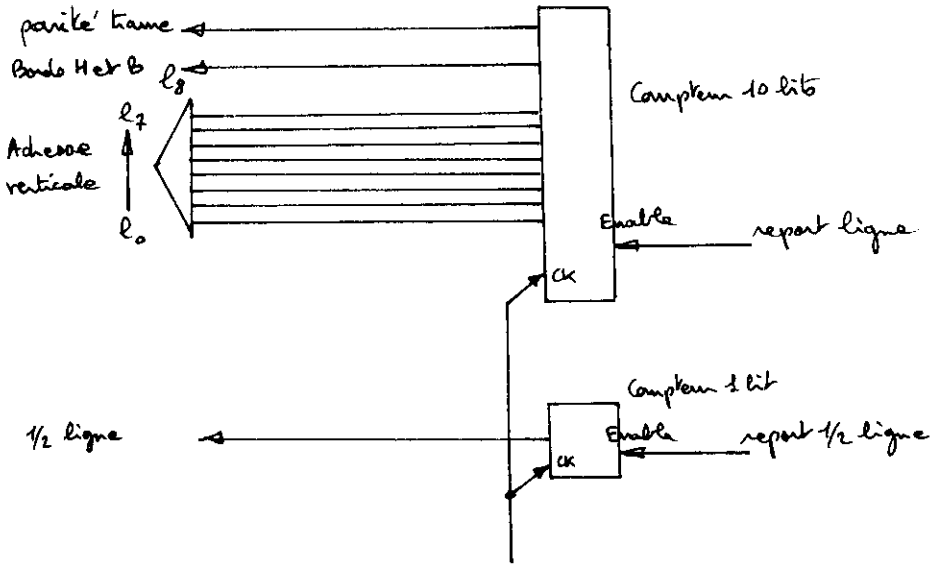


Fig 4.7

Pour la suite des raisonnements, laissons provisoirement tomber le bit de poids le plus fort, et essayons d'obtenir une périodicité de 312,5H pour les autres bits. Notons (m,n) les états de ce compteur, où $\begin{cases} m = \text{valeur décimale du compteur supérieur de 9 bits,} \\ n = \text{bit de demi-ligne (0 ou 1).} \end{cases}$

b)<u>Réalisation de la période 312,5H ou 312H.</u> La solution préconisée est la suivante:

La reconnaissance de la valeur (312,0) fait un Clear (synchrone avec le report de demi-ligne) des 10 bits considérés (Fig.4.8). La croix $\times$ indique l'instant où prend effet le Clear. La figure montre que dans tous les cas, 312,5H séparent 2 croix consécutives.

De plus, une solution simple pour supprimer l'entrelacement des

trames consiste à ignorer le bit de demi-ligne, et de faire un Clear lorsque l'on reconnaît (312,X). On se trouve alors toujours dans le cas de la colonne de droite, et 2 croix sont séparées de 312H.

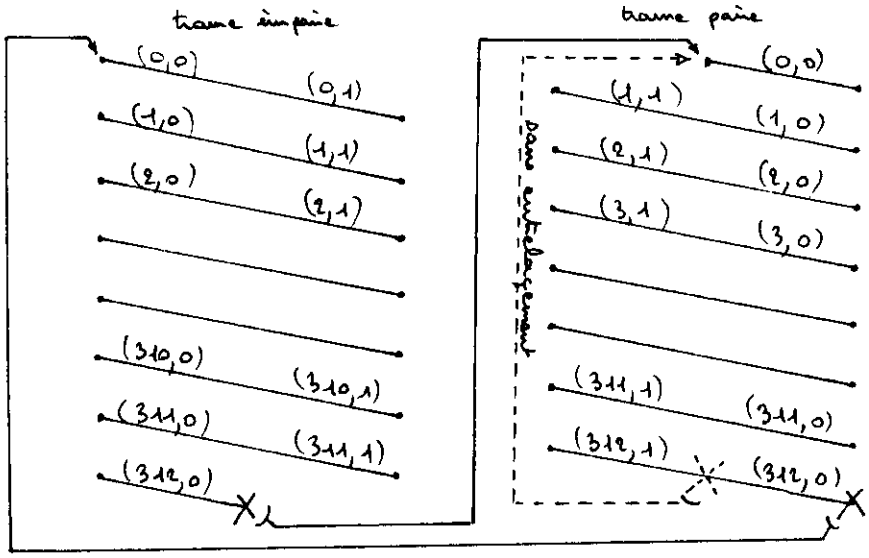


Fig I.8 : états du compteur vertical

c)Obtention de la parité trame. Le bit de parité trame (11ème bit) peut être obtenu de 2 manières:

-On peut utiliser le report constitué par la reconnaissance de la valeur (312,0) pour faire basculer ce dernier bit.

-On peut enregistrer la valeur du bit faible (de demi-ligne) lors de chaque top ligne puisqu'elle est différente d'une trame à l'autre.

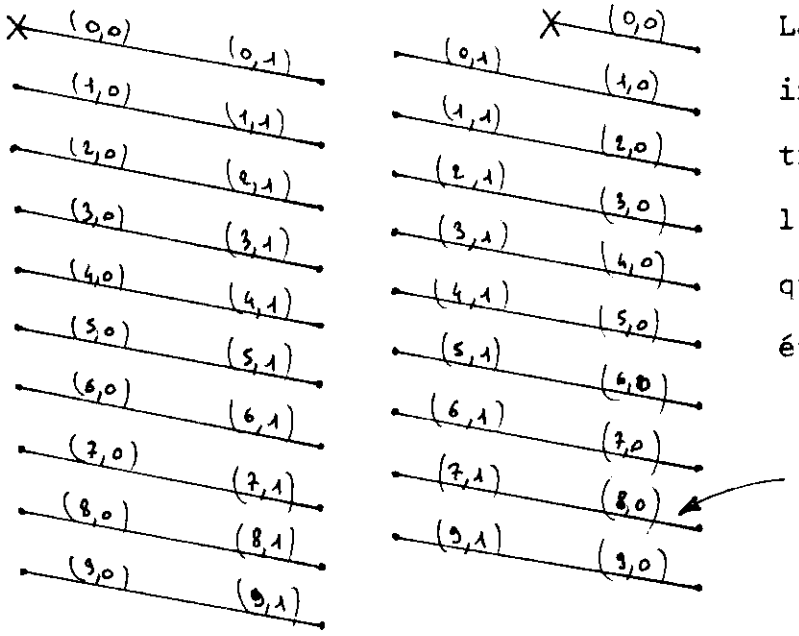
Cette dernière solution est moins sensible aux parasites éventuels puisque la valeur de ce bit est alors actualisée plus souvent. Ce bit sera donc "1" lors des trames dessinées ci-dessus à gauche, et "0" pour celle de droite. Ce sera donc exactement le bit de poids faible (et non son inverse) si l'on compte de haut en bas, puisque la ligne (0,0) de droite est au dessus de la ligne (0,0) de gauche.

d)Modification du séquencement lors du top trame. Ce système

possède encore un inconvénient: la reconnaissance du top de synchro trame est différente lors des trames paires et des trames impaires. Ceci peut se résoudre de 2 façons:

-<u>1ère solution:</u> On remplace les reconnaisseurs sur le compteur vertical par des reconnaisseurs sur un petit compteur additionnel incrémenté toutes les demi-lignes, et démarrant sur le Clear du grand compteur. Il suffit de 4 bits puisqu'il y a $15x\frac{H}{2}$ de durée de top trame, plus un état de blocage. Ceci a l'avantage, du point de vue de l'implantation du circuit intégré, de supprimer de longues connexions de reconnaisseurs et de les remplacer per une partie plus compacte. Elle a par contre l'inconvénient de rajouter 4 éléments de compteur.

-<u>2ème solution:</u> On peut modifier le séquencement du compteur au voisinage du top trame, de la façon suivante: si la partie supérieure (de 9 bits) a une valeur inférieure à 8, alors le report d'entrée de cette partie sera calculé sur l'étage inférieur (bit de demi-ligne), même s'il arrive au milieu de ligne. On aboutit à la séquence suivante en début de trame:



La trame ici à gauche est inchangée, mais dans la trame à droite, c'est l'état (8,X) qui ne dure que H/2, et non plus l'état (0,X).

Fig 4.9

Dans les 2 méthodes le top trame est constitué de:

-pré-égalisation: (0,X)+(1,X)+(2,0)

-serration: (2,1)+(3,X)+(4,X)

-post-égalisation: (5,X)+(6,X)+(7,0)

Le plus simple pour générer le top trame est de reconnaître les 2 sortes de périodes suivantes:

-enveloppe du top trame: de (0,X) à (7,0)

-serration: de (2,1) à (4,X)

Nous avons utilisé la deuxième solution qui occupe tout de même moins de place.

4.4-SCHEMA D'ENSEMBLE

Ce séquencement doit maintenant être adapté à la contrainte suivante: De même que pour le compteur horizontal, l'état 0 doit être le haut de l'image intéressante. Il faut donc décaler tous les états d'une quantité fixe.

En particulier, le changement de trame se fait en reconnaissant la valeur (272,0) pour sauter à (472,0). Le saut est en effet de $2^9-312=200_{(10)}$.

Les autres valeurs se déduisent par le même décalage. Il faut en outre reconnaître la ligne (19,X) -c.à.d. (491,X) après décalage- pour forcer la vidéo à blanc. En effet, les téléviseurs l'utilisent comme référence du niveau du blanc [11] (appellée "ligne test" sur les figures qui suivent).

Pour terminer, il faut utiliser les sorties inverses des numéros de lignes et du bit de parité trame pour que la ligne 0 de l'image graphique se situe en bas de l'écran.

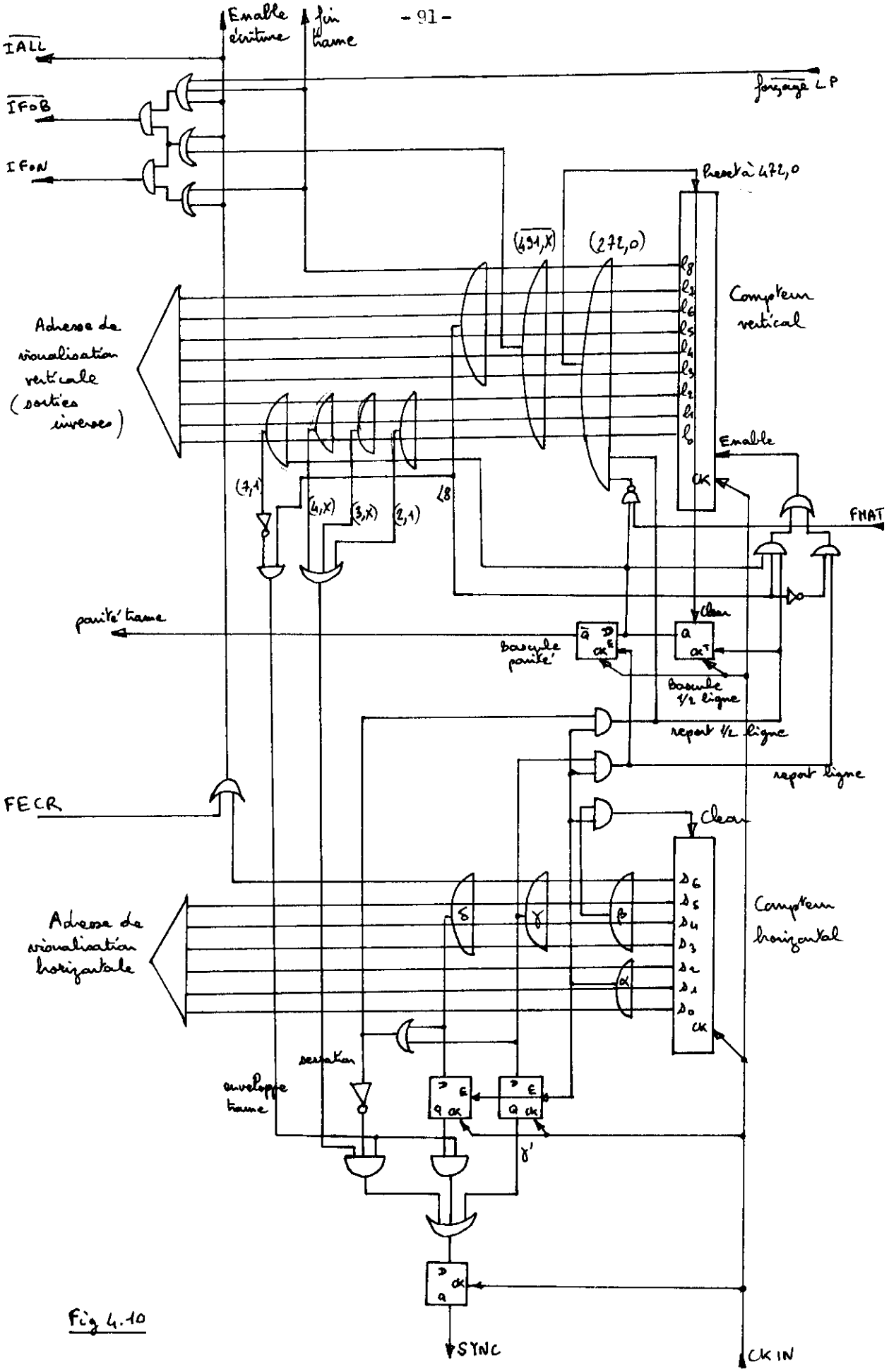

Enable écriture
fin trame
IALL
forçage LP
IFoB
IFoN
Reset à 472,0
(431,X)
(272,0)
Adresse de visualisation verticale (sorties inversées)
Compteur vertical
Enable
CK
(1,1)
(4,X)
(3,X)
(2,1)
L8
FMAT
parité trame
bascule parité
Clear
bascule 1/2 ligne
report 1/2 ligne
report ligne
FECR
Clear
Adresse de visualisation horizontale
Compteur horizontal
δ
γ
β
α
sensation
enveloppe trame
γ'
SYNC
CK IN

Fig 4.10

Le schéma de principe du compteur entier est donné Fig.4.10. Sur la sortie SYNC est intercalée une bascule D pour "nettoyer" ce signal. Il s'ensuit un décalage de T sur le cadrage horizontal qui ne gêne en rien. Les sorties s_0-s_5, parité trame et l_0-l_7 sont utilisées pour l'adressage de la mémoire, mais aussi en entrée des registres "X light pen" et "Y light pen".

4.5-MAQUETTE DE SIMULATION MSI

Les reconnaisseurs sont réalisés à l'aide de mémoires mortes 256x4 bits SFC 71301 (Fig.4.11).

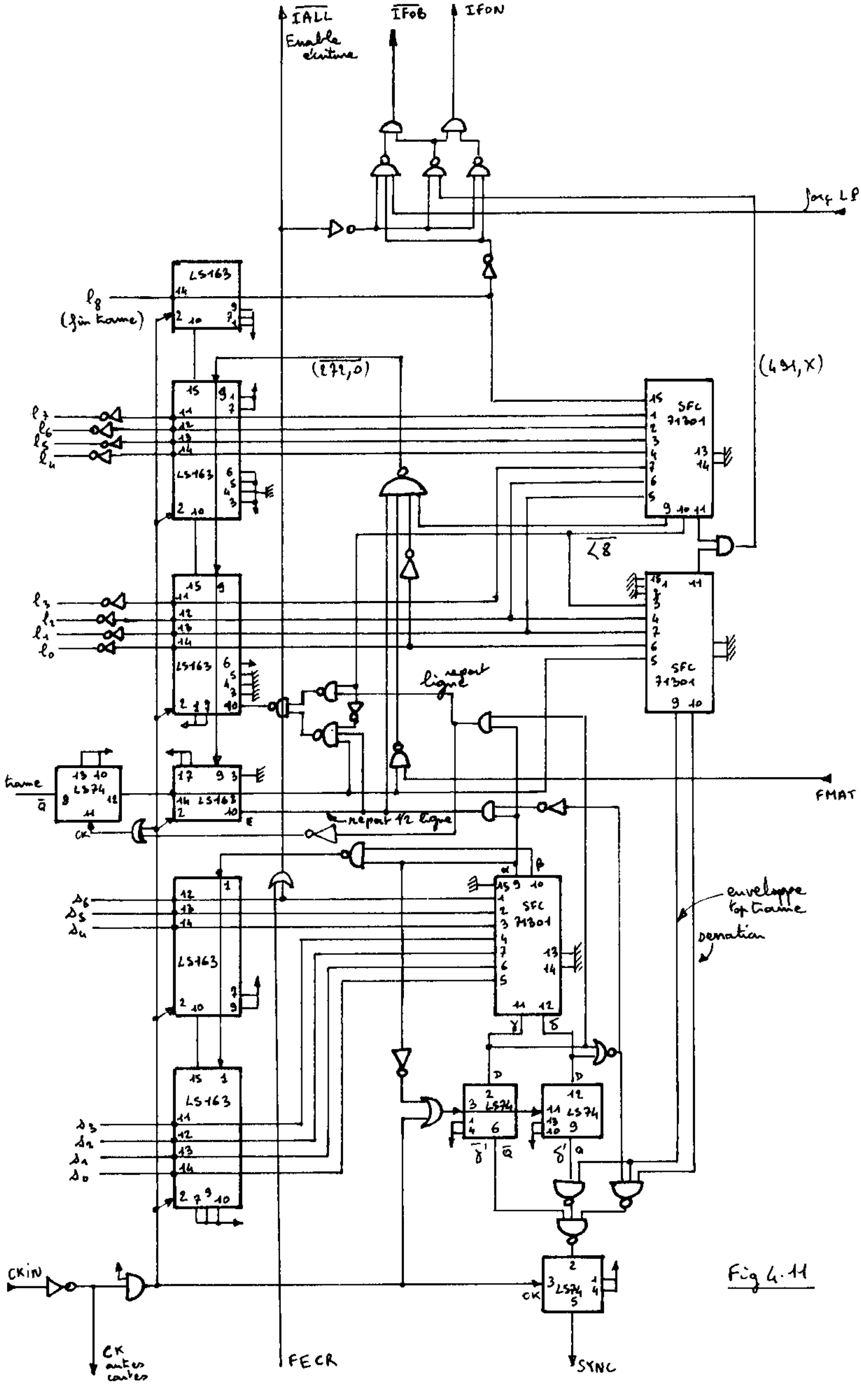

IALL
Enable écriture
IFOB
IFON
Larg LP
LS163
l_8 (fin trame)
(272,0)
(491,X)
SFC 71301
l_7
l_6
l_5
l_4
l_3
l_2
l_1
l_0
L8
repart ligne
trame
LS74
FMAT
repart 1/2 ligne
CK
s_6
s_5
s_4
s_3
s_2
s_1
s_0
enveloppe top trame
dénation
CKIN
CK autres cartes
FECR
SYNC

Fig 4.11

4.6-IMPLANTATION DU COMPTEUR HORIZONTAL

C'est un compteur UP synchrone avec Clear synchrone. Un étage peut être réalisé à partir de briques 4, 8, 10. Les bascules décalant γ et δ sont des briques 4, 7, la bascule "nettoyant" SYNC est une brique 4.

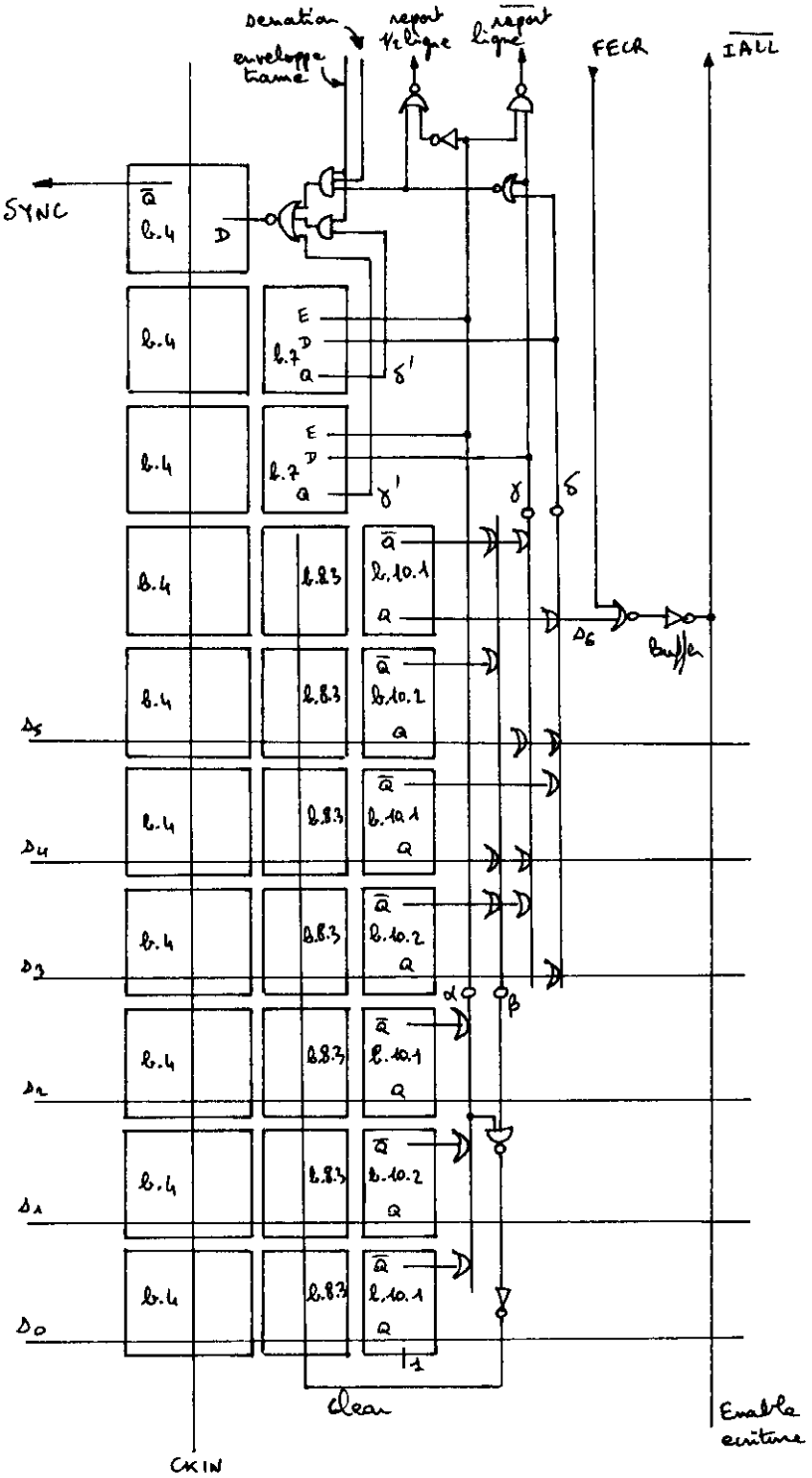


Fig 4.12

4.7-IMPLANTATION DU COMPTEUR VERTICAL

C'est un compteur UP avec Preset. Nous utiliserons aussi les briques 4, 8, 10. La bascule de demi-ligne est constituée de briques 4 et 8, celle calculant la parité trame de briques 4 et 7.

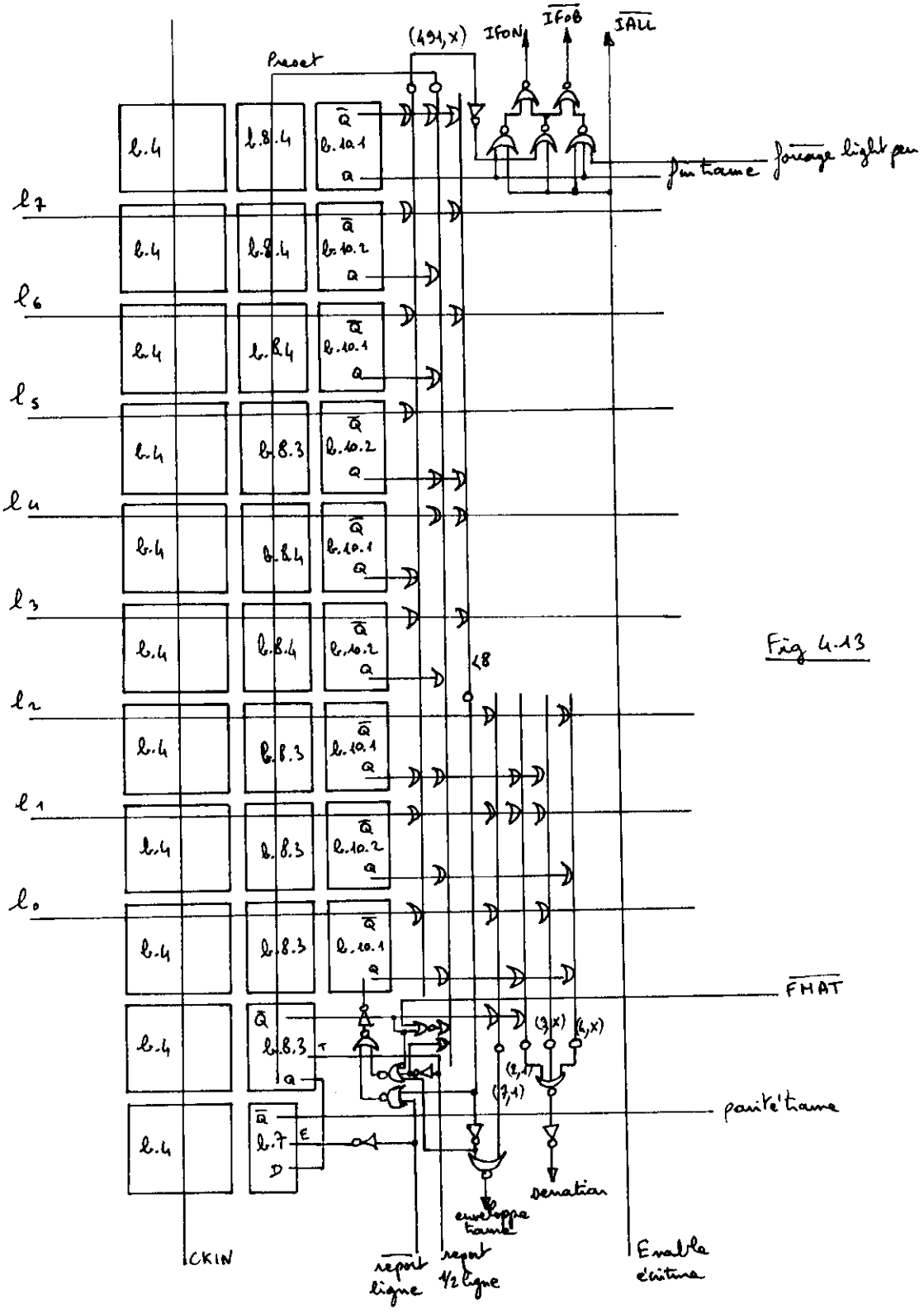


Fig 4.13

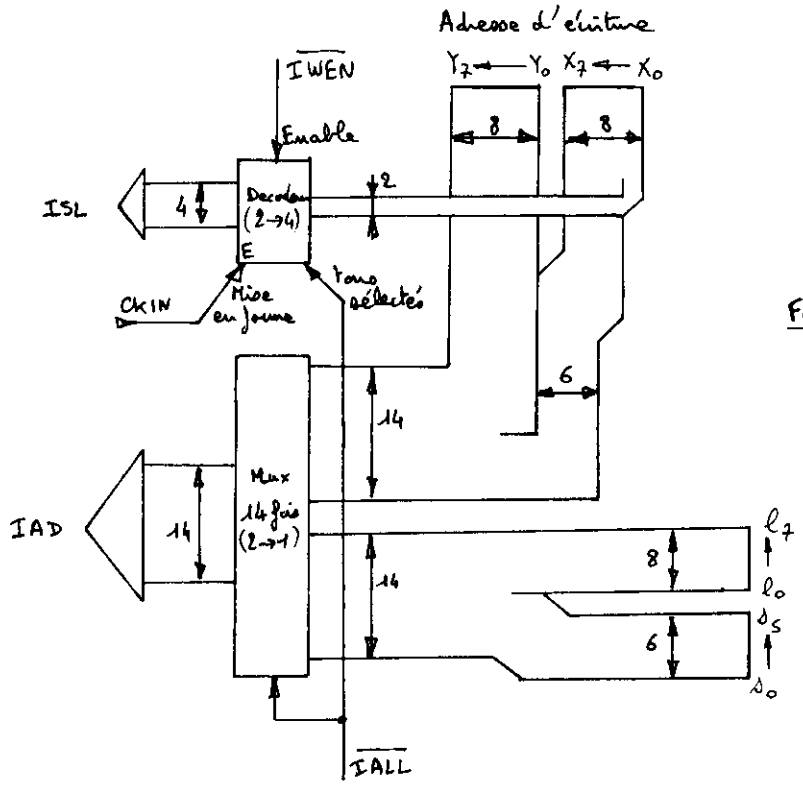


Fig 4.14 : 256x256 et format inférieur

Compteur de visualisation

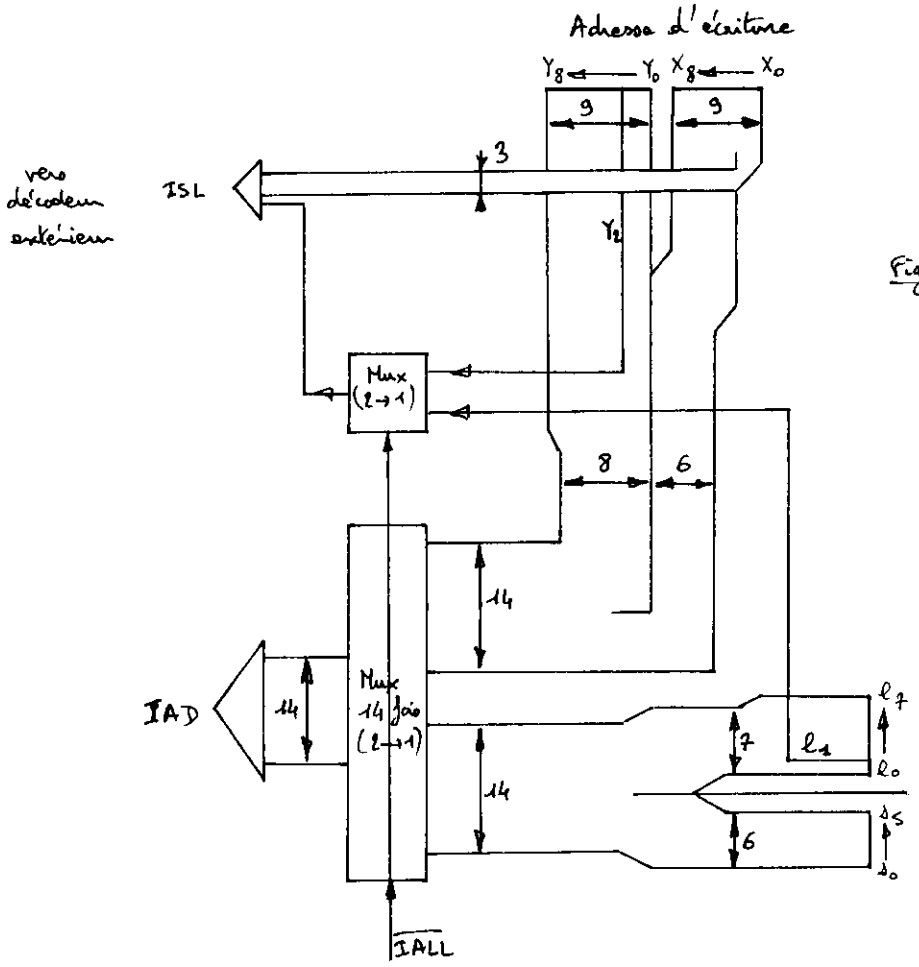


Fig 4.15 : 512x512

Compteur de visualisation

B-SIGNAUX DE CONTROLE DE LA MEMOIRE D'IMAGE

4.8-POSITION DU PROBLEME

Nous disposons d'une adresse de visualisation sur 15 bits:

$S_0, S_1, S_2, S_3, S_4, S_5$, parité trame, $l_0, l_1, l_2, l_3, l_4, l_5, l_6, l_7$,

et d'une adresse d'écriture sur 18 bits:

$X_0, X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, Y_7, Y_8$.

Nous devons: (voir §2.4 et 2.5)

-multiplexer ces adresses suivant l'écriture ou la visualisation en fonction de $\overline{IALL}$.

-associer des adresses d'écriture différentes aux adresses de visualisation, en fonction de l'entrée FMAT.

-répartir ces adresses entre les sorties IAD et ISL.

-multiplexer les parties haute et basse sur les sorties IAD en fonction de l'entrée CKIN.

En laissant de côté le multiplexage entre parties haute et basse, et en représentant les 2 formats sur 2 schémas différents, la répartition entre IAD et ISL est indiquée par les figures 4.14 et 4.15. Dans le format 512x512, rappellons que ISL3 sert de selection de moitié de la mémoire et change toutes les 2 lignes lors de la visualisation pour des raisons de rafraîchissement (voir §2.6.d).

4.9-SORTIES ISL DE SELECTION DE BOITIER MEMOIRE

Les sorties ISL sont donc le multiplexage en fonction de l'entrée FMAT des 2 fonctions réalisées sur les figures 4.14 et 4.15. Ceci peut se symboliser par la figure 4.16.

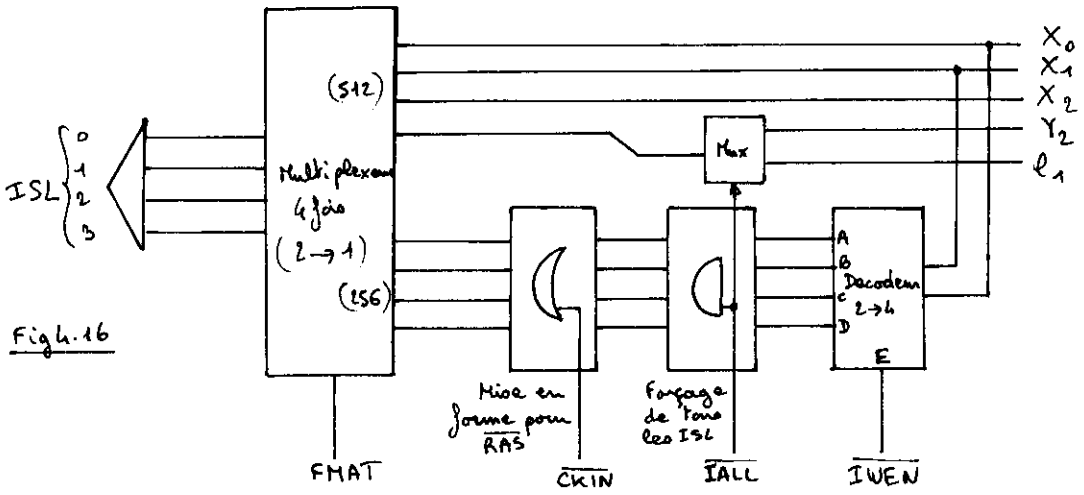


Fig4.16

4.10-MULTIPLEXEURS D'ADRESSES MEMOIRES ET SORTIES IAD

Le problème restant consiste à répartir les adresses sur les sorties IAD en fonction des 3 commandes FMAT,$\overline{\text{IALL}}$,CKIN. Pour faire abstraction du multiplexage suivant CKIN entre parties haute et basse de l'adresse, nous nommerons les sorties IADi par R0,R1,..R6,C0,C1..C6 en considérant que ces sorties sont au nombre de 14.

L'attribution des différentes adresses est obtenue après remplissage du tableau 4.1 de la façon suivante (Ce tableau doit être considéré comme la concaténation d'un tableau 512x512 -à gauche- et d'un tableau pour les autres formats -à droite- qui auraient en commun la colonne 3. Mais il n'y a pas de correspondance à chercher entre le groupe des colonnes 1 et 2 et celui des colonnes 4,5,6,7):

1°)On remplie tout d'abord la colonne 3 avec les noms des adresses de visualisation disponibles.

2°)On peut alors remplir les colonnes 1 et 5,6,7 en fonction des correspondances écriture/visualisation définies par le chapitre 2 (schémas d'application).

3°)On place alors un attribut ISL aux adresses d'écriture

envoyées sur les sorties ISL.

Tableau 4.1

Colonne 1	Colonne 2	Colonne 3	Colonne 4	Colonne 5	Colonne 6	Colonne 7
Adresses d'écriture en 512x512	Affectation des adresses pour IAD en 512x512	Adresses de visualisation disponibles	Affectation des adresses sur IAD en 256 x 256	Adresses d'écriture en 256x256	Adresses d'écriture en 128x128	Adresses d'écriture en 64x64
Y_8	C_0	l_7	C_0	Y_7	Y_7	Y_7
Y_7	C_1	l_6	C_1	Y_6	Y_6	Y_6
Y_6	C_2	l_5	C_2	Y_5	Y_5	Y_5
Y_5	C_3	l_4	C_3	Y_4	Y_4	Y_4
Y_4	C_4	l_3	C_4	Y_3	Y_3	Y_3
Y_3	C_5	l_2	C_5	Y_2	Y_2	Y_2
Y_2 (ISL)		l_1	C_6 (a)	Y_1	Y_1	
Y_1	R_6 (d)	l_0	R_6 (b)	Y_0		
Y_0	C_6 (c)	parité trame				
X_8	R_0	Δ_5	R_0	X_7	X_7	X_7
X_7	R_1	Δ_4	R_1	X_6	X_6	X_6
X_6	R_2	Δ_3	R_2	X_5	X_5	X_5
X_5	R_3	Δ_2	R_3	X_4	X_4	X_4
X_4	R_4	Δ_1	R_4	X_3	X_3	X_3
X_3	R_5	Δ_0	R_5	X_2	X_2	X_2
X_2 (ISL)				(ISL) X_1	(ISL) X_1	
X_1 (ISL)				(ISL) X_0		
X_0 (ISL)						

Il reste alors à placer les noms des 14 sorties RO,..R6,CO..C6 dans les colonnes 2 et 4.

4°)On place alors C6 en a et R6 en b d'après les montages d'application 64x64 et 128x128.

5°)On doit placer une adresse Ci en c pour ne pas gêner le rafraîchissement en 512x512 puisque la parité trame ne change que tous les 20ms. Plaçons-y C6 en même temps que nous plaçons R6 en d pour minimiser les différences entre les colonnes 2 et 4 et donc aussi la complexité des multiplexeurs.

6°)Il reste alors à compléter les colonnes 2 et 4 par des sorties Ci en haut et des sorties Ri en bas (en face des adresses de visualisation qui tournent le plus rapidement, pour un bon

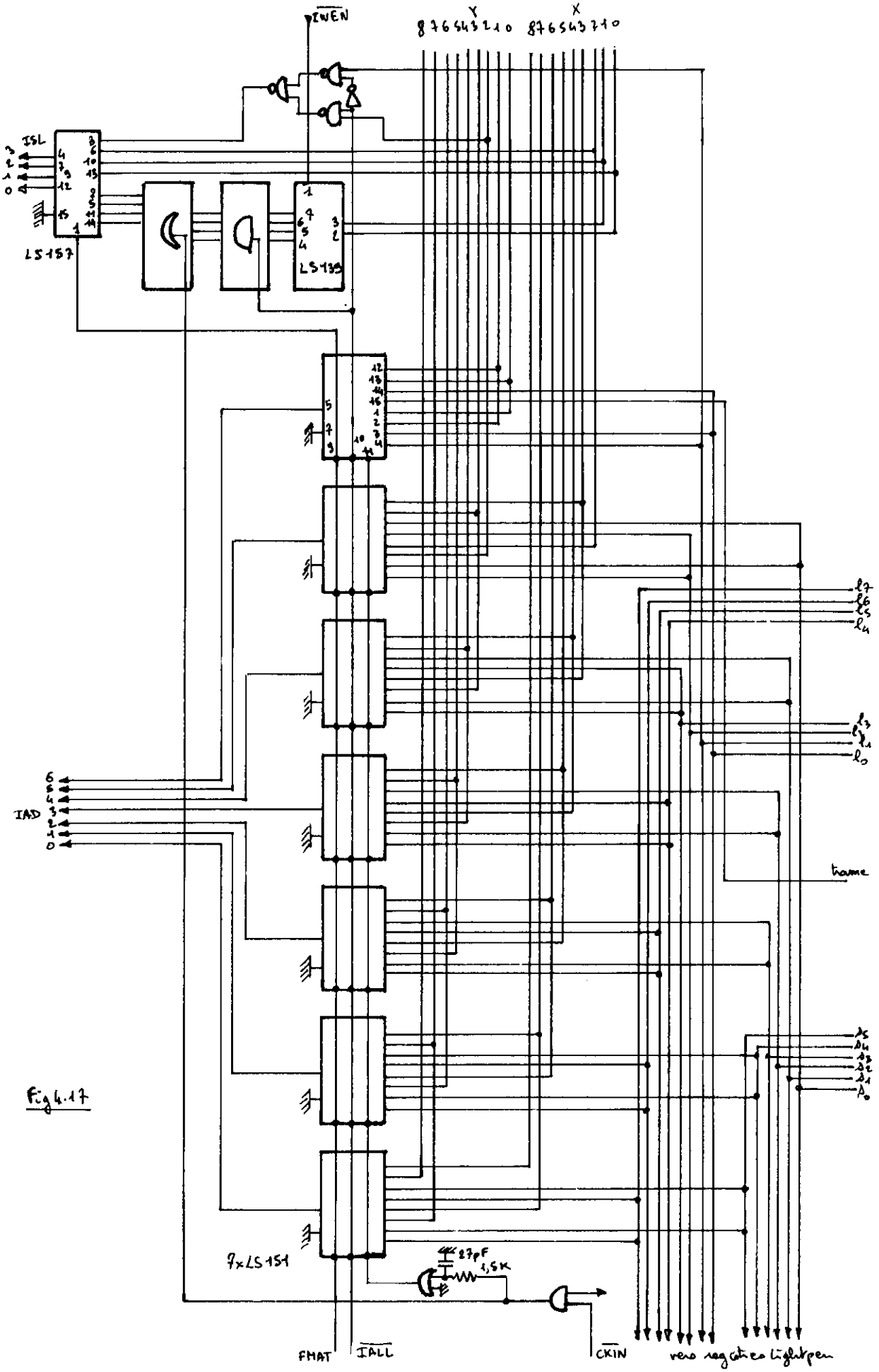

IWEN
Y
8 7 6 5 4 3 2 1 0
X
8 7 6 5 4 3 2 1 0
ISL
LS157
LS139
IAD
6
5
4
3
2
1
0
9xLS151
87pF
1,5K
frame
FMAT
IALL
CKIN
vers registres lightpen

Fig 4.17

rafraîchissement). En mettant les mêmes numéros dans les colonnes 2 et 4, on fait tomber à 6 le nombre d'entrées de presque tout les multiplexeurs générant IAD. Par exemple, la sortie IAD3 présente alternativement s_2 , l_4 , Y_5 , Y_4 , X_5 , X_4 .(Fig.4.21).

4.11-MAQUETTE DE SIMULATION MSI

Son schéma (Fig.4.17) qui est très simple bien que comportant beaucoup de fils est la traduction de la figure 4.16 et du tableau 4.1. La commande des multiplexeurs entre parties haute et basse de la mémoire se fait sur CKIN légèrement retardé. En effet, ces multiplexeurs sont bien plus rapides que ne seront les buffers de sorties sur IAD pour le circuit intégré. Les adresses basses risquent donc de disparaître trop tôt, avant d'avoir été échantillonnées par les mémoires sur le front avant de $\overline{RAS}$.

4.12-SCHEMAS D'IMPLANTATION

a)<u>Commandes de ISL.</u> Les sorties ISL0,1,2 sont calculées en 2 couches à partir de X_0,X_1,X_2. Il en est de même pour ISL3 si FMAT est bas.

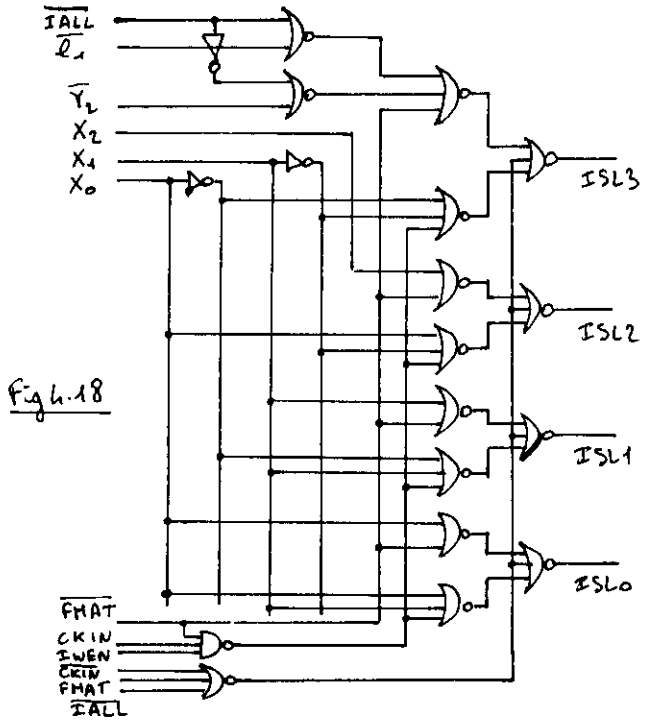


Fig 4.18

Ce calcul est un simple multiplexage avec 2 possibilités de forçage:

-forçage à 0 pour le $\overline{RAS}$ de tous les boîtiers en visualisation si FMAT est bas.

-forçage à 1 pour mise en forme du $\overline{RAS}$ d'un seul boîtier

en écriture si FMAT est bas.

Si FMAT est haut, le calcul de ISL3 est plus complexe puisque le multiplexage entre l_i et Y_i est fait séparément, pour des raisons d'implantation (voir Fig.4.23 et 4.24).

b)Structure du multiplexeur commandant IAD. La logique de ce multiplexeur est simple. Par contre, son implantation sur le circuit intégré est plus délicate. Il faut éviter de faire circuler de longues connexions, tout en évitant de rajouter des couches logiques qui ralentiraient les signaux.

On peut remarquer que pour une sortie IAD donnée, la différence entre les 2 formats ne réside que dans un décalage des adresses d'écriture (mis à part la sortie IAD6). Ceci suggère la géométrie de la figure 4.19. Dans notre réalisation, nous avons confondu les 2 multiplexeurs écriture/visualisation et colonnes/rangées, ce qui permet de n'occuper que la largeur de 7 liaisons aluminium pour ces 2 multiplexeurs.

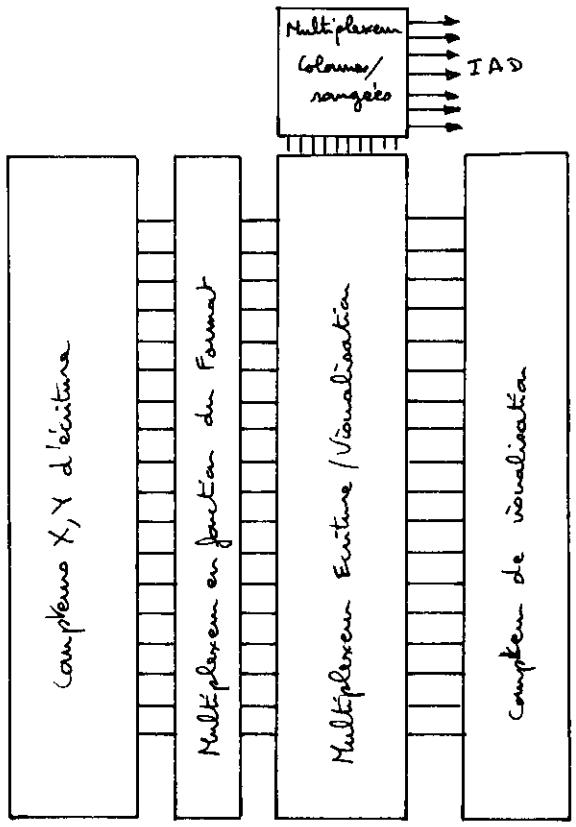


Fig 4.19

Cette géométrie est satisfaisante sur le plan temporel puisque le temps critique est celui de commutation entre adresses de rangées et adresses de colonnes (τ_1). Or c'est celui de commutation du tout dernier multiplexeur et du buffer. De plus, d'après les spécification des mémoires, τ_2 peut s'annuler, et même atteindre -10ns pour certains modèles [12], ce qui correspond, dans le cas où CKIN a un rapport cyclique de 1, où FMAT est bas, et où $\overline{CAS}$ dure T/4, à τ_1 max=$\frac{547}{4}$≃135ns. Mais on peut (§2.5) modifier à volonté $\overline{RAS}$ et $\overline{CAS}$. On peut aussi décaler le signal $\overline{Load}$ du registre à décalage.

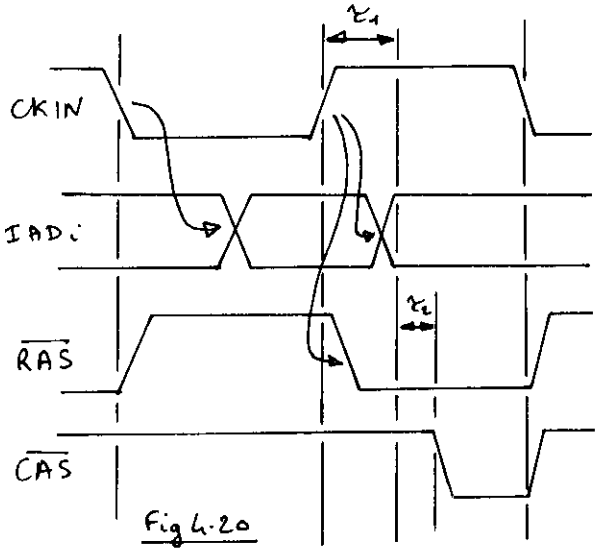


Fig 4.20

Nous avons réalisé l'implantation de la façon suivante: Fig.4.21 4.22, 4.23, et 4.24.

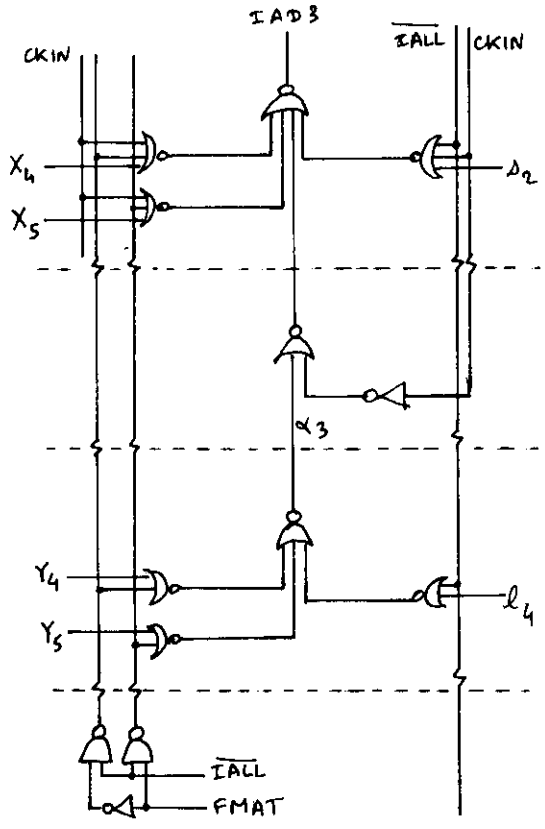


Fig 4.21
Seule la commande de IAD3 est représentée

-Il n'est requis qu'une seule connexion verticale par fil d'adresse.

-La commutation entre adresse de rangée et de colonne ne concerne que 2 couches logiques puisque α_i est déjà prêt avant que CKIN monte.

Le fil IAD6 comporte une entrée supplémentaire:

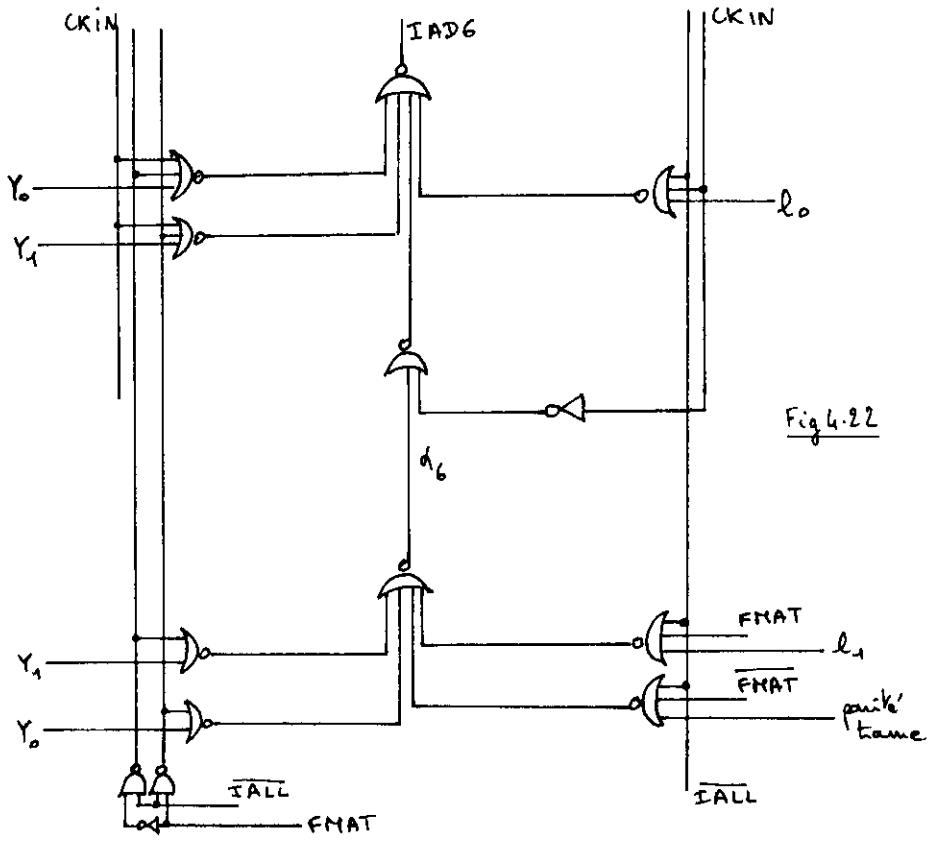


Fig 4.22

c)Implantation de la partie donnant les adresses verticales.

Elle est indiquée en Fig.4.23. En A, on a utilisé ($\overline{IALL}$.FMAT) au lieu de $\overline{IALL}$. Ceci n'est pas gênant puisque si FMAT est bas, on n'utilise pas la sortie du NOR (voir Fig.4.18). On peut rajouter un fil horizontal pour amener $\overline{IALL}$ si on n'a pas la place de mettre cet inverseur.

Ce schéma n'utilise pas de briques car un multiplexeur dépend beaucoup de l'endroit où il se place. De plus, celui-ci doit être optimisé en surface et en vitesse. Au moment de l'implantation, il se peut qu'une brique s'avère facile à utiliser, mais ceci n'est pas sûr au niveau où nous en sommes.

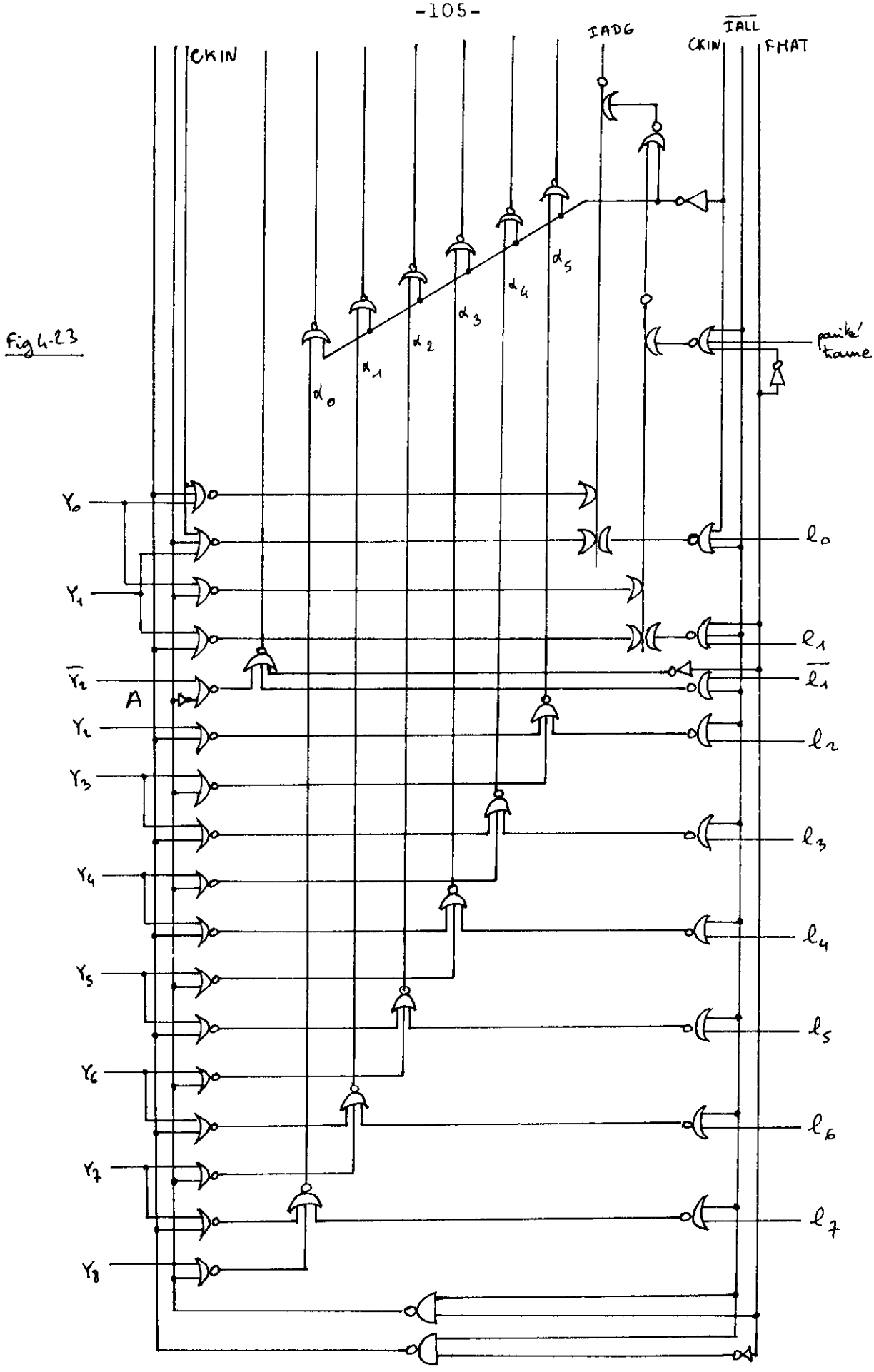

CKIN
IADG
IALL
CKIN
FMAT
α_0
α_1
α_2
α_3
α_4
α_5
parité' trame
Y_0
Y_1
$\overline{Y_2}$
A
Y_2
Y_3
Y_4
Y_5
Y_6
Y_7
Y_8
ℓ_0
ℓ_1
$\overline{\ell_1}$
ℓ_2
ℓ_3
ℓ_4
ℓ_5
ℓ_6
ℓ_7

Fig 4-23

d)Implantation de la partie donnant les adresses horizontales et les sorties ISL.

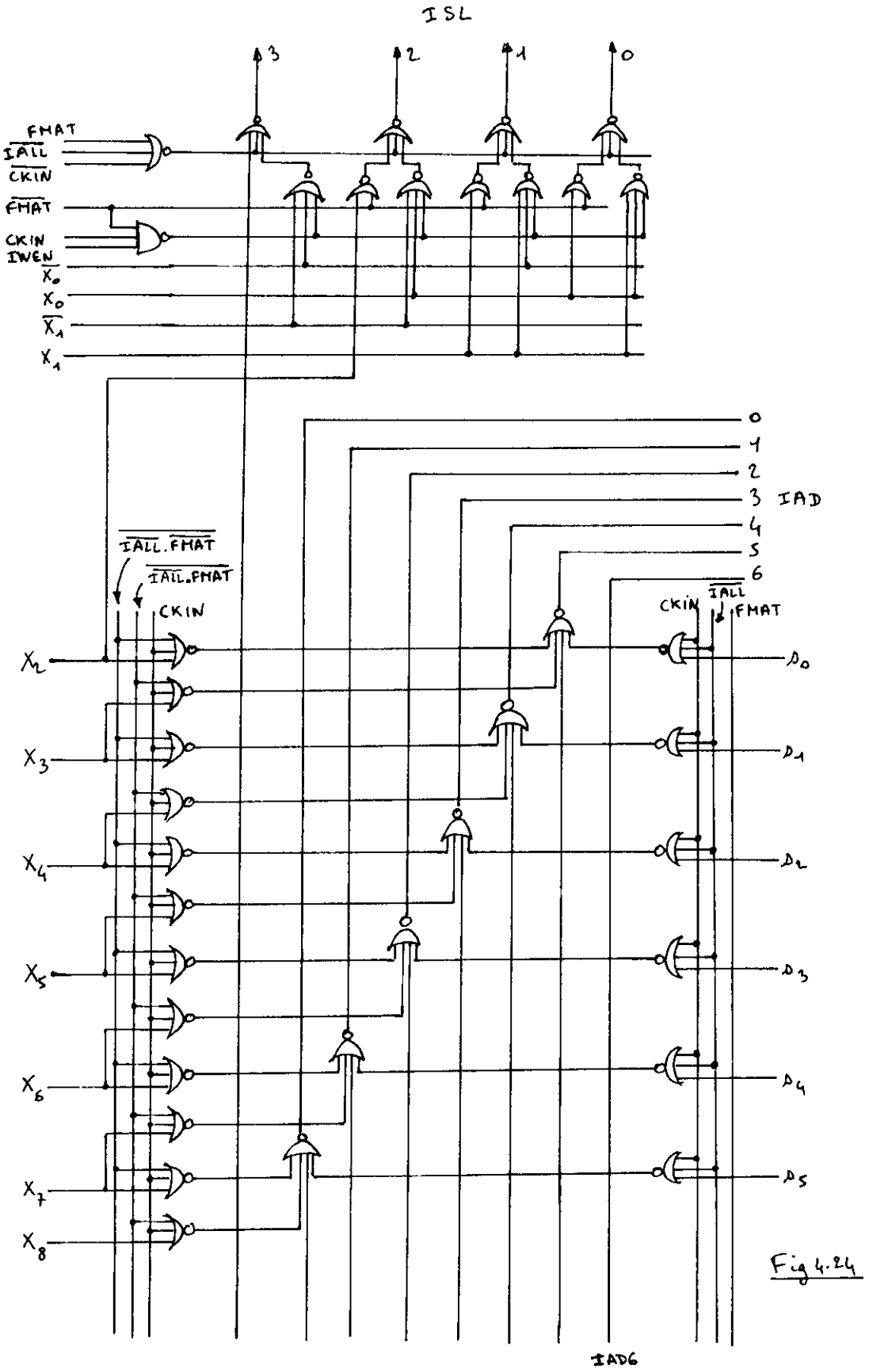


Fig 4.24

5-GENERATEUR DE VECTEURS

5.1-Position générale du problème de la génération de vecteurs sur une grille.

5.2-Algorithme classique (de BRESENHAM).

5.3-Position "hardware" du problème.

5.4-Solution 1: utilisation de BRMs.

5.5-Solution 2: compteur p-tuple.

5.6-Solution 3: BRM ou compteur p-tuple, normalisé.

5.7-Solution 4: compteur modulo q.

5.8-Solution 5 (retenue): compteur p-tuple modulo q.

5.9-Equivalence de cette solution avec l'algorithme de BRESENHAM; "qualités" des vecteurs.

5.10-Symétrisation du rôle des entrées ΔX et ΔY.

5.11-Initialisation.

5.12-Circuits complémentaires; adaptation à la notice.

5.13-Schéma de principe.

5.14-Maquette de simulation MSI.

5.15-Implantation sur le circuit intégré.

a)Latchs $|\Delta X|$ et $|\Delta Y|$.

b)Multiplexeur à 3 entrées.

c)Registre.

d)Compteur UP/DOWN.

e)Forçages sur $|\Delta X|$ et $|\Delta Y|$.

f)Décodage de la direction et contrôle des sorties ENX et ENY du générateur.

g)Petit compteur 2 bits avec ses reconnaisseurs.

h)Génération des pointillés.

i)Structure d'une tranche de 2 bits.

j)Générateur complet.

5.1-POSITION GENERALE DU PROBLEME DE LA GENERATION DE VECTEURS SUR UNE GRILLE

L'écran est constitué d'un ensemble $\mathcal{E}$ de n^2 points situés sur les intersections des lignes d'une grille. Ces points peuvent être "noirs" ou "blancs". Générer un vecteur d'origine (X_o, Y_o) et de composantes $(\Delta X, \Delta Y)$, c'est faire passer un sous-ensemble $\mathcal{S}$ des points de $\mathcal{E}$ à un état choisi, par exemple "blanc". $\mathcal{S}$ doit être tel qu'il "suggère le mieux possible" à l'observateur l'existence d'un segment de droite entre les points (X_o, Y_o) et $(X_o+\Delta X, Y_o+\Delta Y)$.

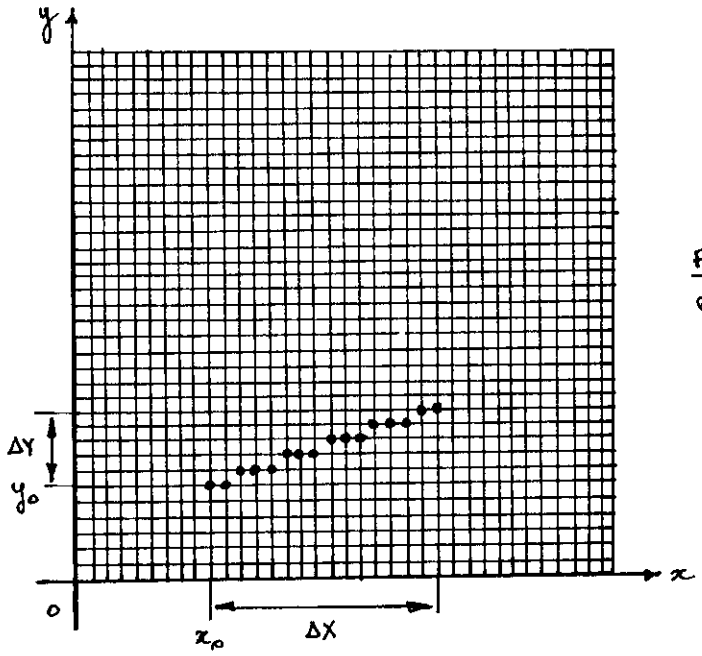


Fig 5.1 : écran et exemple de vecteur

Dans tout ce qui suit, nous considèrerons toujours pour simplifier que $X_o = Y_o = 0$ et que $\Delta X \gg \Delta Y \gg 0$ (vecteurs dans le premier octant).

5.2-ALGORITHME CLASSIQUE (DE BRESENHAM). [16], [17]

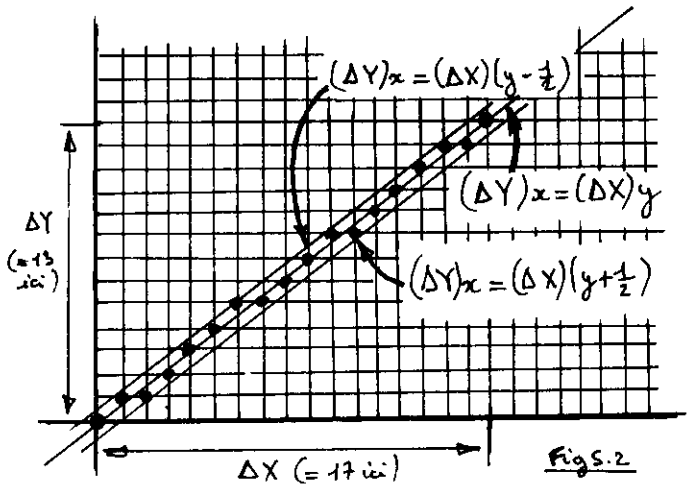


Fig 5.2

L'équation de la droite à approcher est $(\Delta Y)x=(\Delta X)y$. Nous décidons de placer un point dans chaque colonne, à une distance à cette droite, comptée sur la verticale, inférieure ou égale à 1/2 maille de la grille. Dans le cas où 2 points sont à une distance 1/2 au-dessus et au-dessous, nous prendrons le point au-dessus. Tous les points ainsi définis sont situés dans la zone:

$$\begin{cases}(\Delta Y)x \geqslant (\Delta X)(y-\frac{1}{2}) \\ (\Delta Y)x < (\Delta X)(y+\frac{1}{2})\end{cases}$$

La méthode évitant les multiplications et les divisions est la suivante: on part de x=y=0. On incrémente x à chaque pas, et y seulement si nécessaire. Pour cela, on considère l'expression $s=(\Delta Y)x-(\Delta X)(y+\frac{1}{2})$. A chaque pas, on évalue cette expression avec le nouveau x et le précédent y. Si elle est devenue positive, c'est à dire si $(\Delta Y)x \geqslant (\Delta X)(y+\frac{1}{2})$, alors y doit être incrémenté. Or l'expression s s'obtiend très simplement, il suffit de partir de $-\frac{\Delta X}{2}$, d'ajouter ΔY à chaque fois que l'on incrémente x, et de soustraire ΔX à chaque fois que l'on incrémente y. On s'arrête quand x a atteind la valeur ΔX:

```
début x:=0; y:=0; s:=-ΔX/2;
tantque x<ΔX faire
        point(x,y):=noir; x+:=1; s+:=ΔY;
        si s≥0 alors y+:=1; s-:=ΔX fsi
        fait
fin
```

Remarques:

1°)On a toujours $-\Delta X \leqslant s < \Delta Y$. s se code donc sur le même nombre de bits que ΔX et ΔY.

2°)Il n'y a jamais d'incrémentation de y sans incrémentation de x quand $\Delta X \geqslant \Delta Y$. C'est une "qualité" des vecteurs.

3°)A chaque pas, on obtiend un nouveau point. La boucle n'est répétée que $\Delta X+1$ fois, qui est le nombre de points du vecteur.

5.3-POSITION "HARDWARE" DU PROBLEME

Les n^2 points de $\mathcal{E}$ sont des bits d'une mémoire $\mathcal{M}$, adressée par 2 registres x et y (de p bits si $n=2^p$). Tracer un vecteur correspond à actualiser les registres x et y et à procéder à une écriture dans la mémoire à chaque période d'un signal d'horloge CK.

Le générateur de vecteurs est donc un automate qui admet en entrée les paramètres ΔX et ΔY ainsi qu'un signal d'horloge. Il délivre en sortie 2 signaux d'horloge CKX et CKY qui commandent l'incrémentation des registres x et y (qui sont donc en fait des compteurs).

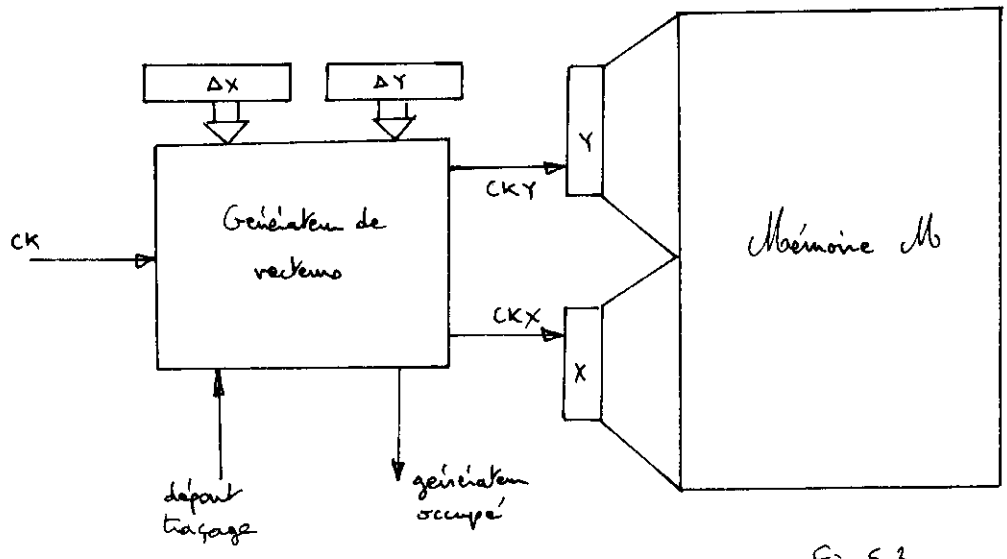


Fig 5.3

Ce générateur de vecteurs peut se découper en 2 parties:

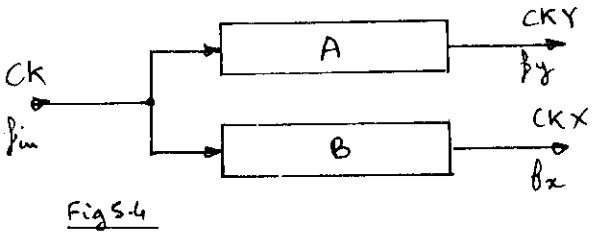


Fig 5.4

où les f_i représentent des fréquences: $\begin{cases} f_y = \frac{\Delta Y}{K} f_{in} \\ f_x = \frac{\Delta X}{K} f_{in} \end{cases}$

$K \geqslant \sup(\Delta X, \Delta Y)$; $f_{in} = \frac{1}{T_o}$

T_o période de CK.

Les 2 "boîtes" A et B sont donc des multiplicateurs de fréquence par $\frac{p}{K}$ ($p \leqslant K$), et nous utilisons en fait l'équation paramétrique de la droite (le temps t étant le paramètre):

$$\begin{cases} y = t f_y = t \frac{\Delta Y}{K} f_{in} \\ x = t f_x = t \frac{\Delta X}{K} f_{in} \\ 0 \leqslant t \leqslant K T_o \end{cases}$$

Le tracé du vecteur prend K périodes de l'horloge CK.

Les différents générateurs de vecteurs se distinguent par la réalisation du multiplicateur de fréquence par $\frac{p}{K}$ et une de leurs caractéristiques est donc K, le nombre de périodes que prend le tracé.

5.4-SOLUTION 1: UTILISATION DE BRMs [18], [19], [20]

Un BRM ("Binary Rate Multiplier") est un compteur synchrone dont le nombre de bits b est celui sur lequel est codé p (Fig.5.5).Si le i^e bit de p est à "1", alors à chaque fois que l'étage de rang (b-i-1) du compteur passe à 1, il est émis une impulsion en sortie.

Comme un étage de rang α du compteur oscille à la fréquence $f_{in}/2^{\alpha+1}$, cela signifie que le ième bit de p valide une sortie à la fréquence $f_{in}/2^{b-i} = f_{in} \times \frac{2^i}{2^b}$. On obtiend donc une décomposition de la fréquence $\frac{p}{2^b}$ en base 2.

Le temps nécessaire pour obtenir p impulsions en sortie est

indépendant de p et est égal à $2^b T_o$. On a donc $K=2^b=\max(\Delta X,\Delta Y)$.

Exemple de BRM: BRM 4 bits. Les poids faibles du compteur sont à droite, ceux de p à gauche. Seul le calcul de la composante de rang 2 du signal est représenté.

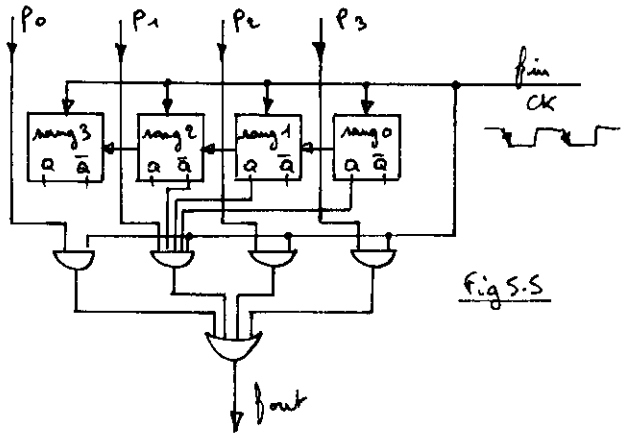


Fig 5.5

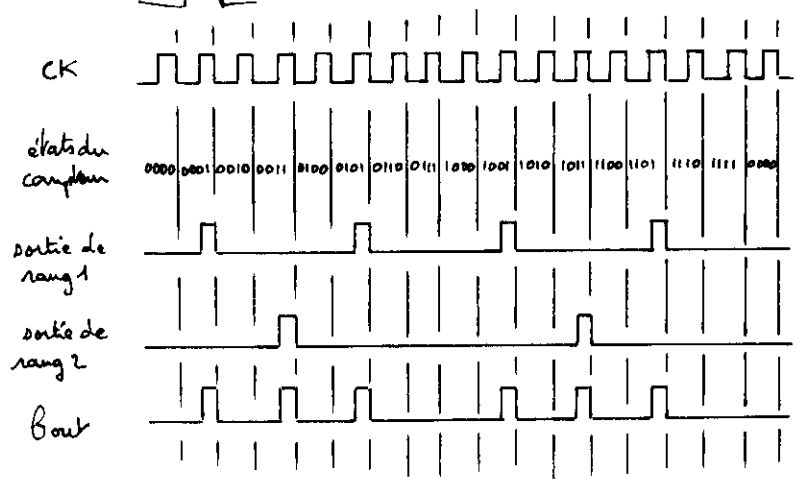


Fig 5.6 :
Exemple :
$b=4$
$p=6=0110_{(2)}$

5.5-SOLUTION 2: COMPTEUR P-TUPLE [21]

Un compteur p-tuple est un compteur qui s'incrémente de p à chaque période d'horloge:

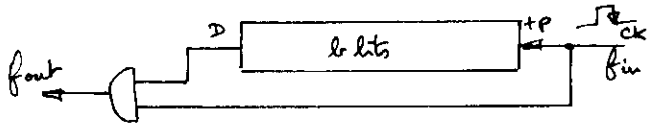


Fig 5.7

D est le dépassement du compteur. Il est à "1" si $N+p\neq(N+p) \bmod 2^b$. N étant son contenu précédent.

Ce compteur peut être réalisé à partir d'un registre non transparent et d'un additionneur (Fig.5.8). Le dépassement (ici report de l'additionneur) arrive p fois plus souvent que le report d'un compteur de même longueur, soit $f_{out}=\frac{p}{2^b} f_{in}$. Il faut 2^b impulsions

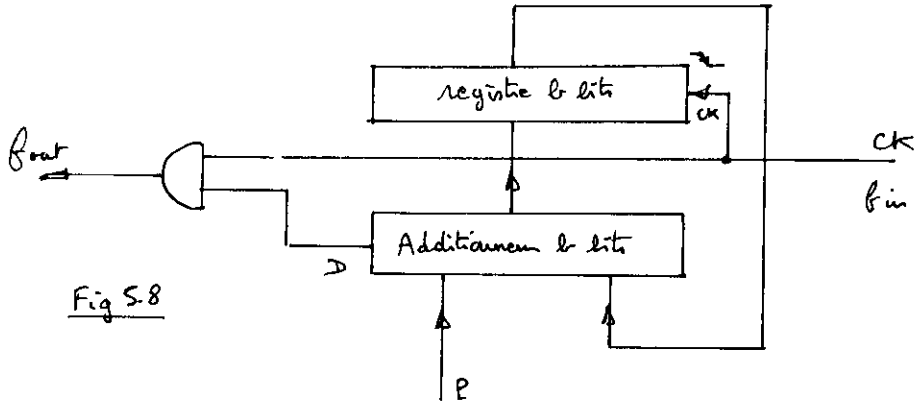


Fig 5.8

en entrée pour en obtenir p en sortie. Une périodicité du registre indépendante de p est $2^b T_o$ puisque $(\sum_1^{2^b} p) \bmod 2^b = 0 \ \forall p$.

Soit q le quotient de 2^b par p; le nombre de périodes de CK entre 2 impulsions de sortie ne peut être que q ou q+1. En effet, après une impulsion de sortie, le contenu du registre est compris entre 0 et p-1, donc après q impulsions d'entrée supplémentaires, il est compris entre qp et $qp+(p-1) = \lceil (q+1)p \rceil - 1$. Donc, comme $qp \leqslant 2^b \leqslant \lceil (q+1)p \rceil - 1 < qp+p$, le dépassement est obtenu après q ou q+1 impulsions suivant le cas. Ceci montre que, contrairement au BRM, les impulsions de sortie sont le mieux réparties possible (compte tenu du fait que dans un système digital, les impulsions de sortie sont forcément alignées sur une impulsion d'entrée).

Si le compteur est initialisé avec la valeur 0, la première impulsion de sortie arrivera après q ou q+1 impulsions d'horloge, et vu la périodicité de $2^b T_o$, la pième impulsion de sortie arrivera à la 2^bième impulsion d'horloge exactement.

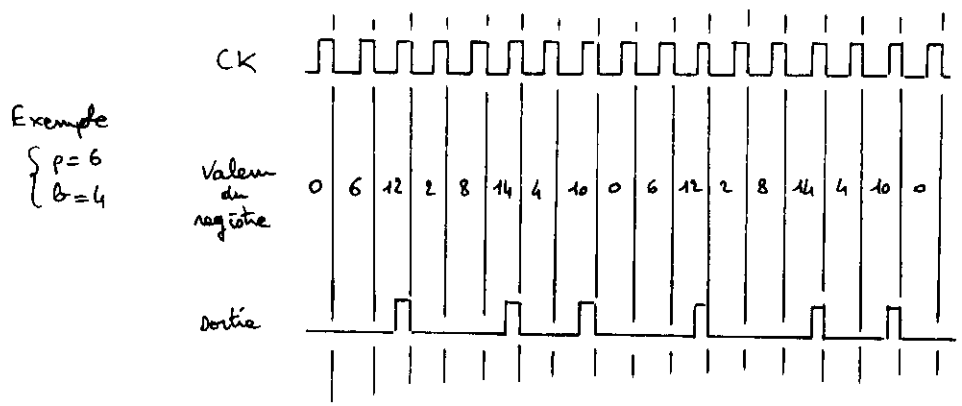


Fig 5.9

Les sorties seront mieux cadrées avec une valeur initiale du registre égale à $\frac{2^b}{2}=2^{b-1}$.

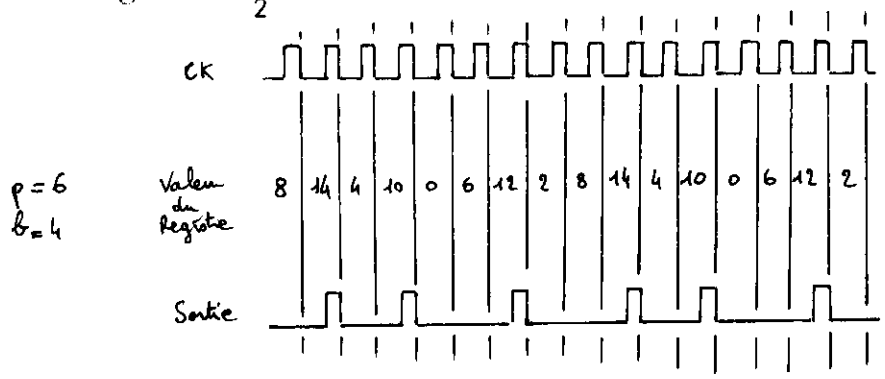


Fig 5.10

De même que pour les BRMs, on a $K=2^b$.

5.6-SOLUTION 3: BRM OU COMPTEUR P-TUPLE, MAIS NORMALISE

L'inconvénient des 2 systèmes précédents est que dans le cas de petits vecteurs, on s'impose un temps de tracé de 2^b périodes, à cause de la rigidité du nombre de bits du montage. On peut chercher à "normaliser" ΔX et ΔY, en leur faisant subir un décalage pour les cadrer à gauche, en simulant ainsi une valeur de b plus faible. On a alors $K=\inf(2^\beta)$ tel que $2^\beta \geqslant \Delta X$ et $2^\beta \geqslant \Delta Y$. Cette méthode a été proposée par SUCHARD et a été utilisée par BERNARDY [18].

5.7-SOLUTION 4: COMPTEUR MODULO Q

Un compteur (un-uple) compte de -q à -1. Pour cela, il suffit de réaliser un chargement à la valeur -q lorsque l'on reconnaît la valeur -1 (report de sortie):

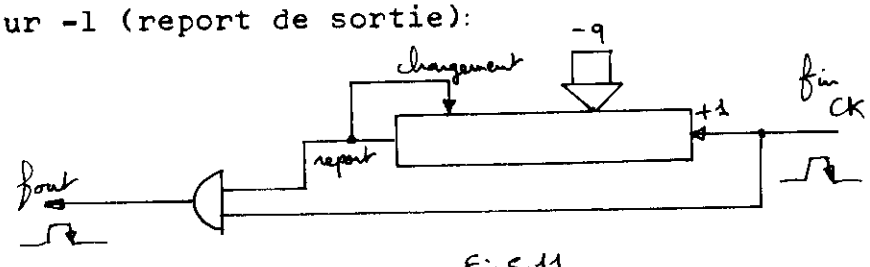


Fig 5.11

Le report apparaît q fois moins souvent que le CK. On a donc $f_{out}=\frac{1}{q}f_{in}$ avec $0<q<2^b$. Dans ce cas, en utilisant 2 compteurs, on peut obtenir:

$$\begin{cases} f_y=\frac{1}{\Delta X}f_{in} \\ f_x=\frac{1}{\Delta Y}f_{in} \\ K=\Delta X,\Delta Y. \end{cases}$$

5.8-SOLUTION 5 (RETENUE): COMPTEUR P-TUPLE MODULO Q

OBERMAN a proposé [21] de rassembler les solutions 2 et 4 pour réaliser un compteur p-tuple modulo q -ici ΔY-tuple modulo ΔX-. On obtiend ainsi:

$$\begin{cases} f_y=\frac{\Delta Y}{\Delta X}f_{in} \\ f_x=f_{in} \\ K=\Delta X \end{cases}$$

Il faut réaliser un compteur comptant de -q à -1 par sauts de p (Fig.5.12)

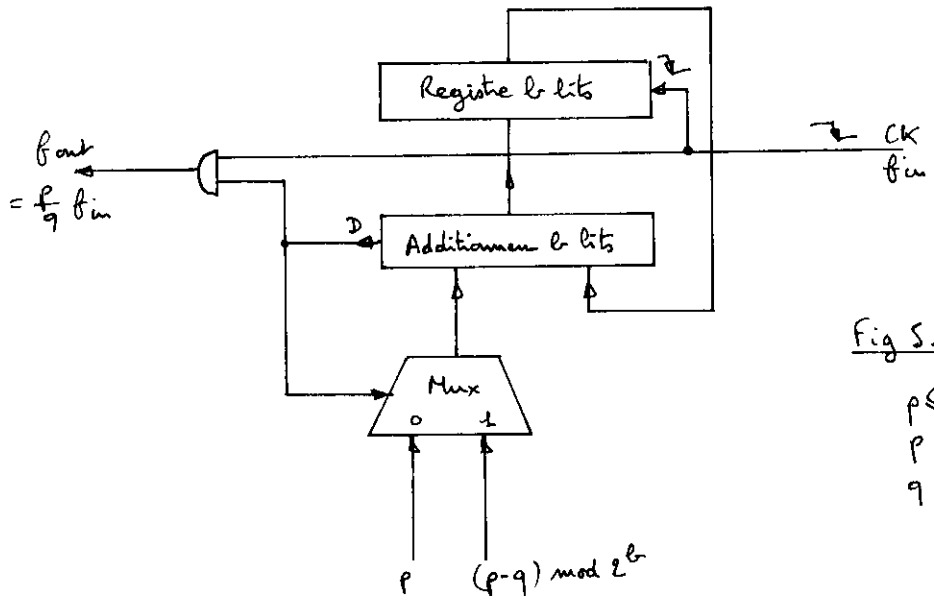


Fig 5.12 :

p ≤ q
p = ΔY
q = ΔX

Cela signifie que q doit jouer ici le rôle que jouait 2^b dans le §5.5. Or cette valeur intervient lors du dépassement. En effet, à ce moment, la valeur réellement ajoutée au contenu du registre est $p-2^b$. Il faut donc maintenant ajouter p-q. Or, puisqu'il y a

dépassement, on perd dans ce cas aussi 2^b. Il faut donc ajouter exactement $p-q+2^b=(p-q) \bmod 2^b$ puisque $p \leqslant q$.

Les valeurs N du registre telles que $-2^b < N < -q$ sont interdites. Il convient de ne pas initialiser le registre à une de ces valeurs.

De la remarque concernant l'initialisation du compteur de la solution 2, et par analogie, nous voyons qu'il faut initialiser le registre à -q/2 (sans se soucier des ereurs d'arrondi car au pire cela ne peut faire que décaler d'une période de CK les impulsions de sortie).

Si l'addition de p crée un dépassement, l'addition de $(p-q) \bmod 2^b$ en crée un aussi à plus forte raison. En effet, $(p-q) \bmod 2^b = p-q+2^b$ puisque $p \leqslant q$ or $p-q+2^b > p$ puisque $q < 2^b$.

Mais, si l'addition de p ne crée pas de dépassement, il se peut que l'addition de p-q en crée un, et cet état est stable, sans être l'état attendu. D'autre part, sur le plan de la réalisation physique, le bouclage additionneur-multiplexeur est instable (Fig.5.13). Ces 2 dernières raisons obligent à interposer sur la commande du multiplexeur une bascule non transparente. Chaque période d'horloge comprendra les phases suivantes:

-forcer à 0 la commande du multiplexeur,

-échantillonner le dépassement de l'additionneur,

-laisser cette valeur échantillonnée commander le multiplexeur,

-échantillonner enfin la sortie de l'additionneur dans le registre.

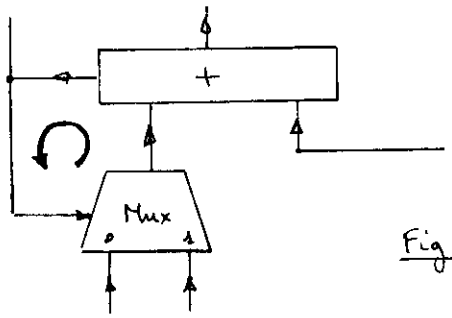


Fig 5.13

Le schéma devient:

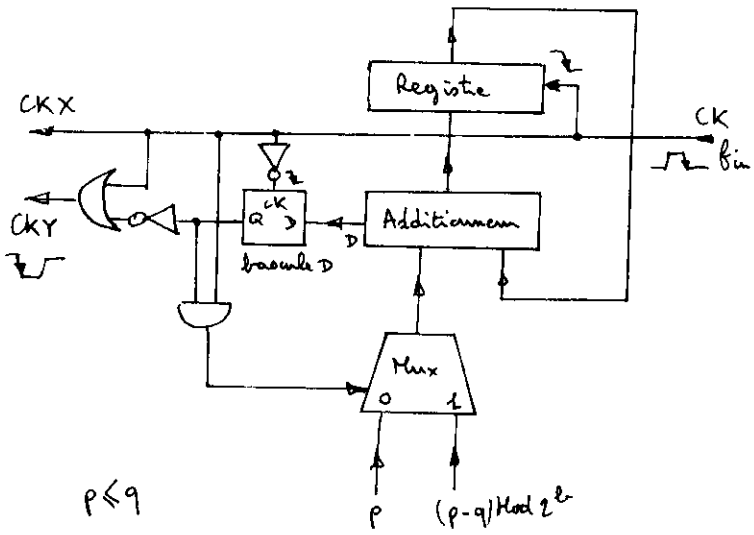


Fig 5.14

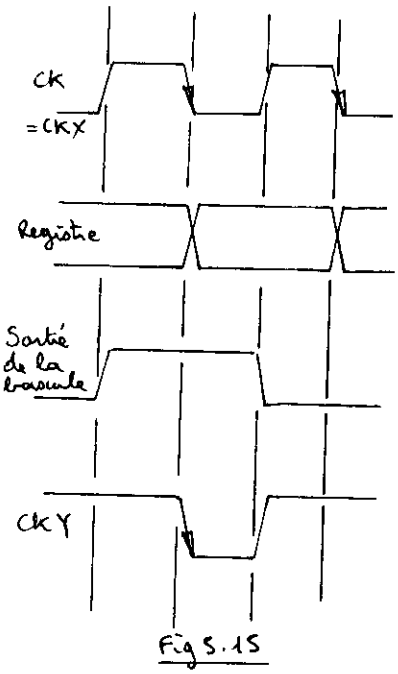


Fig 5.15

5.5-EQUIVALENCE DE CETTE SOLUTION AVEC L'ALGORITHME DE BRESENHAM; QUALITE DES VECTEURS

Si nous appellons s le contenu du registre, nous voyons que cette variable évolue exactement de la même façon que la variable de même nom du §5.2, et que les incréments en X et Y sont obtenus avec les mêmes conditions. Ce montage réalise donc exactement l'algorithme de BRESENHAM (mis à part l'arrêt). Les vecteurs tracés sont les mêmes et en ont donc les "qualités":

-il n'y a jamais d'incrémentation de y sans incrémentation de x,

-à chaque période d'horloge, on obtiend un nouveau point du vecteur. La vitesse est donc optimale.

Mais aussi: tous les points sont compris dans la zone:

$$\begin{cases} (\Delta Y)x \geqslant (\Delta X)(y-\frac{1}{2}) \\ (\Delta Y)x < (\Delta X)(y+\frac{1}{2}) \end{cases}$$

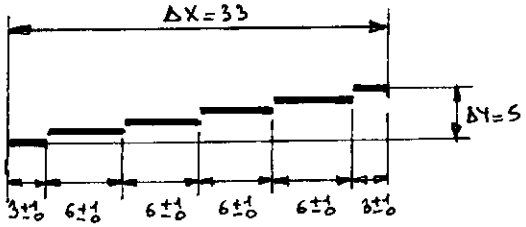
Inversement, nous avons vu entre temps d'autres caractéristiques de ces vecteurs:

-les écarts entre les incréments suivant y diffèrent d'au plus un incrément en x (Fig.5.16).

-la remarque faîte à propos de l'initialisation montre que les segments des extrémités sont la moitié (à 1 près) de ces écarts (ce qui permet un bon enchaînement de 2 fois le même vecteur (Fig.5.16)).

Fig 5.16:

$\Delta X = 6, \Delta Y + 3$



Test d'arrêt: aucun signal interne à ce compteur ne permet de déterminer à quelle phase du traçage on se trouve. L'état du registre n'est pas une indication, puisqu'il peut exister des périodicités différentes de qT_e si p et q ne sont pas premiers entre eux (voir par exemple les figures 5.9 et 5.10). La seule solution pour arrêter le traçage après q impulsions d'entrée est de rajouter un compteur, qui compte par exemple de -q à -1 (par pas de 1).

En fait, il existerait bien une autre solution qui consiste à utiliser X lui-même, comme dans l'algorithme écrit §5.2. Seulement, dans le cas général, le vecteur ne part pas de X=0. Cette solution consisterait donc à enregistrer X (valeur initiale), à lui ajouter ΔX et à comparer ce résultat au X courant. L'ensemble registre (de longueur $\sup(\log_2 X)$) + additionneur + comparateur est plus gros qu'un compteur (de longueur $\sup(\log_2|\Delta X|) < \sup(\log_2 X)$). Mais son principal inconvénient est de nécessiter des connexions nombreuses entre les registres X et $|\Delta X|$. Ce problème est une illustration de ce qui est dit dans l'introduction (chapitre 0).

5.10-SYMETRISATION DU ROLE DES ENTREES ΔX ET ΔY

Nous ne disposons pas encore du générateur de vecteurs de la figure 5.3 dans lequel X et Y ont un rôle symétrique. Pour le réaliser, une solution est de présenter aux entrées du montage de la figure 5.14:

$$\begin{cases} p=\inf(\Delta X,\Delta Y) \\ \text{et} \\ q=\sup(\Delta X,\Delta Y) \end{cases}$$

puis d'aiguiller les signaux de sortie f_{in} et f_{out} vers les compteurs X et Y ou vers Y et X, suivant l'ordre de ΔX,ΔY. Cette solution nécessite un comparateur entre ΔX et ΔY, puis des multiplexeurs aux entrées et aux sorties du montage, commandés par le comparateur.
Une solution moins lourde consiste à faire la remarque suivante: Recopions la figure 5.12 en remplaçant toutes les valeurs par leurs opposées, et en permutant p et q. Nous réalisons un compteur q-tuple modulo p ($q \leqslant p$). Les états du registre sont compris entre 0 et p-1. On génère une impulsion en sortie quand il n'y a pas de dépassement.

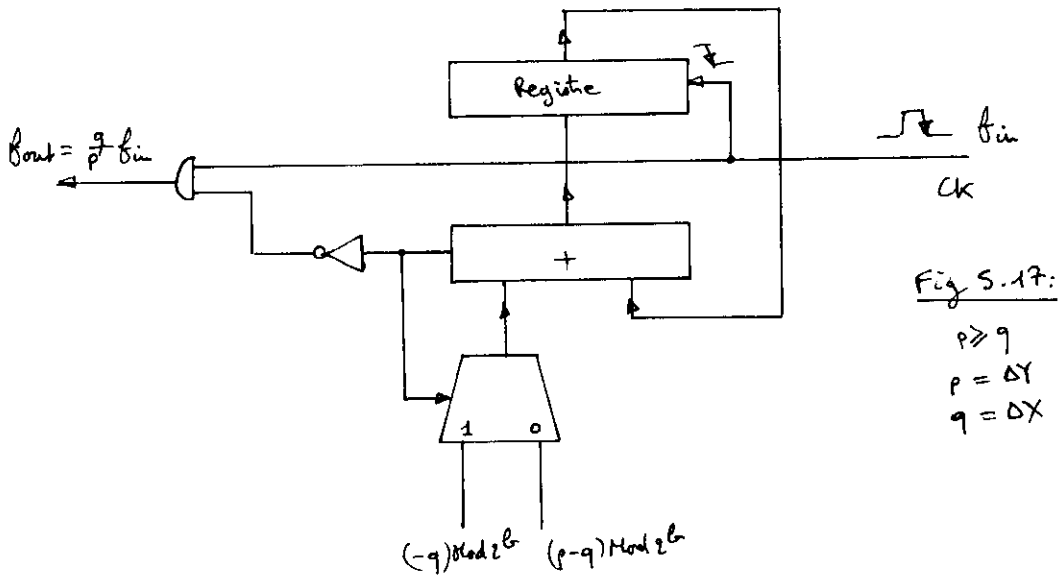


Fig 5.17:
$p \geqslant q$
$p = \Delta Y$
$q = \Delta X$

Toutes les remarques faîtes à propos de la solution 5 se transposent aisément, en particulier:

-l'initialisation du registre doit se faire à p/2=ΔY/2,

-le test d'arrêt se fait grâce à un compteur supplémentaire qui évolue de ΔY à 1 au cours du traçage.

Pour les mêmes raisons qu'au §5.8, on doit intercaler une bascule sur la commande du multiplexeur, et le schéma utilisable devient:

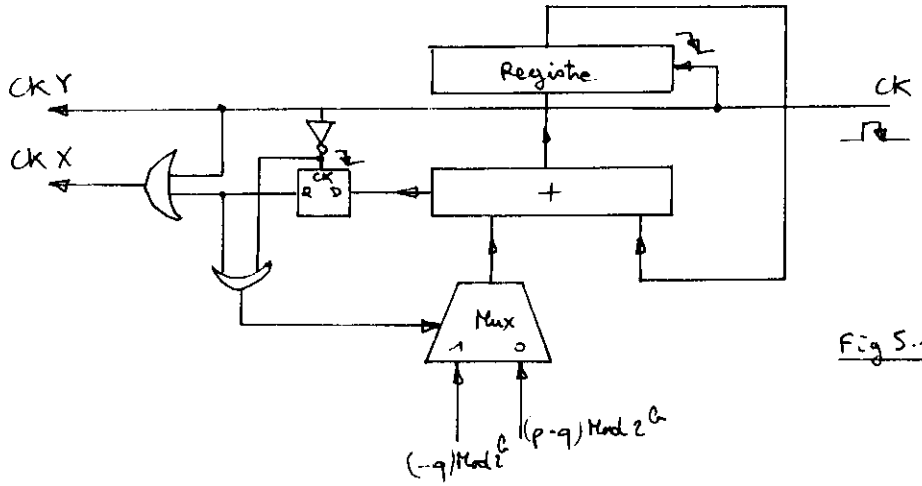


Fig 5.18

L'intérêt de cette transposition est que nous pouvons fondre ensemble les figures 5.14 et 5.18:

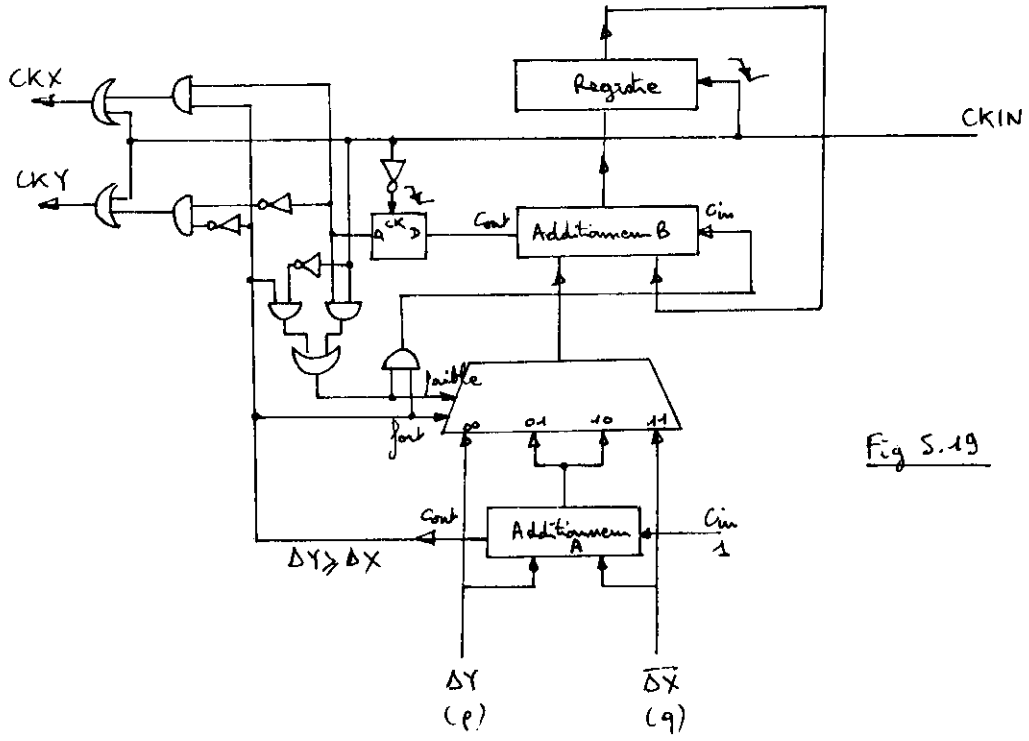


Fig 5.19

en modifiant le point suivant: $(-\Delta X) \bmod 2^b$ est remplacé par $\overline{\Delta X}$.

L'additionneur A dont le report d'entrée est 1 calcule $(\Delta Y - \Delta X) \bmod 2^b$.

Son report de sortie représente donc $\Delta Y \geqslant \Delta X$ (voir §3.7). Ce report de sortie commande le poids fort du multiplexeur qui devient à 3 entrées. Mais dans le cas où l'entrée "11" du multiplexeur est choisie, la sortie de celui-ci est $\overline{\Delta X}$ et non $-\Delta X$. Il faut alors rajouter 1 sur l'additionneur B par son report d'entrée. Ce montage présente l'avantage d'une structure de bus assez simple. La symétrisation est obtenue par l'ajout de quelques portes (à gauche de la figure 5.19) et d'une entrée au multiplexeur.

5.11-INITIALISATION

Nous avons vu que l'initialisation consiste en:

-si $\Delta Y \leqslant \Delta X$: charger $-\Delta X$ dans un compteur UP et dans le registre, puis décaler le registre à droite en forçant le bit de poids fort à 1 (division par 2),

-si $\Delta X \leqslant \Delta Y$: charger ΔY dans un compteur DOWN et dans le registre puis décaler le registre à droite en forçant le bit fort à 0.

ce qui se réalise de la façon suivante: (Fig.5.23) La sortie de l'additionneur B chargera un compteur UP/DOWN en plus du registre. La commande UP/$\overline{\text{DOWN}}$ de ce compteur sera la sortie $\overline{(\Delta Y \geqslant \Delta X)}$ de l'additionneur A. La séquence d'initialisation sera:

1°) Forçage du registre à 0,

2°) Forçage des 2 commandes du multiplexeur à la même valeur, celle-ci étant $\overline{(\Delta Y \geqslant \Delta X)}$,

3°) Enregistrement de la sortie de l'additionneur B dans le registre et le compteur,

4°) Décalage à droite du registre d'un cran (en forçant le bit fort à $\overline{\Delta Y \geqslant \Delta X}$)

Si l'on utilise:

-un registre possédant un Clear synchrone,

-un registre et un compteur UP/DOWN possédant un Load synchrone,

les signaux de commande de l'initialisation peuvent être:

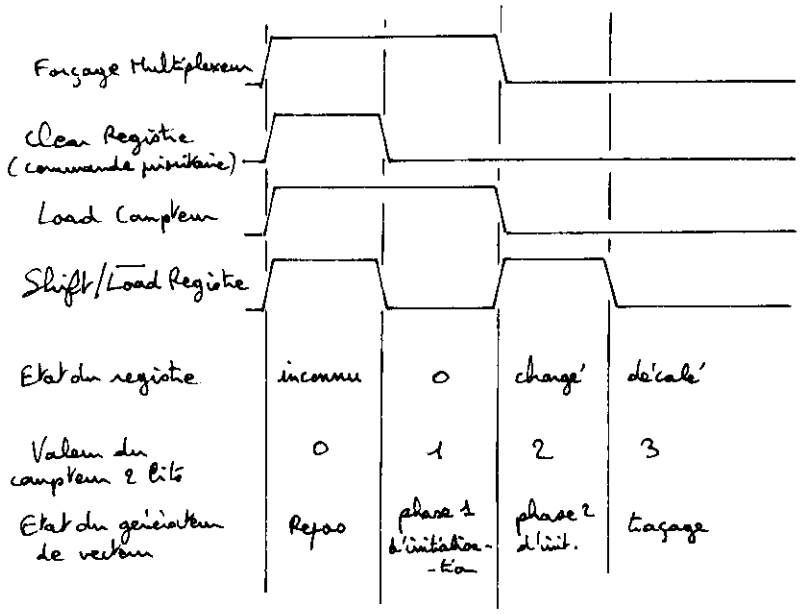


Fig 5.20

Les phases 1 et 2 durent toujours une période d'horloge. La phase 0 dure tant que le générateur est inactif. La phase 3 est la phase de tracé réel du vecteur, la seule pour laquelle les sorties CKX et CKY sont validées.

L'inverse du signal Clear registre est le signal "générateur occupé".

Cette séquence sera commandée par un compteur synchrone annexe de 2 bits, commandée par le même signal d'horloge que le reste du générateur de vecteurs. La reconnaissance des états (-1 et UP) ou (1 et DOWN) du compteur U/D entraînera un Clear synchrone du compteur 2 bits.

Mais en fait, l'état (1 et DOWN) d'un compteur n'est pas pratique à reconnaître. Il est plus simple de faire évoluer le compteur U/D de ΔY à 0 et de $-\Delta X-1$ à -1 en le faisant compter systématiquement une période de plus (pendant la phase 2 de

l'initialisation) alors que la sortie du générateur n'est pas activée. Alors, ceci impose d'initialiser le compteur à -ΔX-1 lorsque $\Delta Y \leqslant \Delta X$, c'est à dire d'éviter le report d'entrée de l'additionneur B lors de l'initialisation (voir Fig.5.23).

Nous avons choisi de valider la sortie "spot" du générateur pendant les phases 2 et 3. Dans tous les cas, ceci signifie que lors de la phase 2 et lors de la première période de la phase 3, on écrit en mémoire, et à la même adresse (celle du point origine du vecteur) puisque les premiers fronts actifs sur les sorties CKX et CKY se placent à la fin de la première période de la phase 3.

Cette validation lors de la phase 2 possède un avantage: si on veut tracer un vecteur pour lequel ΔX=ΔY=0, alors dès la phase 2 le compteur UP/DOWN (en position DOWN) sera chargé de ΔY(=0). Il apparaîtra la reconnaissance de l'état (0 et DOWN) qui fera un Clear du petit compteur 2 bits. On passera donc directement de la phase 2 à la phase 0, tout en ayant bien marqué le point d'adresse (X,Y) en mémoire lors de la phase 2.

Dans le cas général, si m est sup($|\Delta X|$,$|\Delta Y|$), alors la phase 3 dure m périodes, et le temps total de tracé d'un vecteur est (m+3) périodes (à partir du moment où le générateur est activé).

5.12-CIRCUITS COMPLEMENTAIRES, ADAPTATIONS A LA NOTICE

L'insertion de ce générateur dans le circuit intégré "console graphique" demande les adaptations suivantes:

a) Nous nous proposons de tracer des vecteurs tels que:

$$0 \leqslant |\Delta X| \leqslant 255$$

$$0 \leqslant |\Delta Y| \leqslant 255$$

ce qui fixe à 8 bits la largeur de tous les composants du générateur

(b=8).

b) Les entrées sont non pas ΔX et ΔY mais $|\Delta X|$ et $|\Delta Y|$. Les signes de ΔX et ΔY sont précisés par un code de direction de 0 à 7 (figure 2.21). Du point de vue du générateur, seuls $|\Delta X|$ et $|\Delta Y|$ sont intéressants. Les signes de ΔX et ΔY n'interviennent que comme commande UP/DOWN des compteurs d'adresse d'écriture X,Y.

De même, les éventuelles nullités de ΔX ou ΔY (codes de direction 0,2,4, ou 6) interviennent comme commandes de $\overline{\text{Enable}}$ de ces compteurs X,Y. Le décodage de la direction se fait comme indiqué figure 5.21.

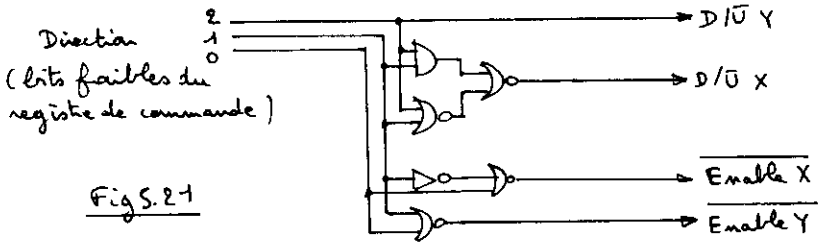


Fig 5.21

Mais ces éventuelles nullités intéressent le générateur de vecteurs. En effet, imaginons que $|\Delta X|$ soit 100 et $|\Delta Y|$ soit 5 et que nous traçions un vecteur dans la direction 2 ($\Delta X=0$, $\Delta Y>0$). Le compteur X ne bougera pas, mais le générateur recevant $|\Delta X|$ et $|\Delta Y|$ considèrera que $|\Delta X| \geqslant |\Delta Y|$ et délivrera ses 5 incréments en Y étalés sur 100 périodes d'horloge. Pour accélérer dans ce cas, il faut forcer l'entrée $|\Delta X|$ du générateur à 0. Ce forçage à 0 ne doit pas intervenir lorsque l'on trace un vecteur de "direction privilégiée" puisqu'alors on doit avoir 100 incréments en Y (voir §2.11).

c) Dans le cas de vecteur de direction privilégié, il suffit de forcer les 2 sorties CKX et CKY à être identiques à l'horloge CKIN pour obtenir le vecteur défini en 2.11.

d) Les entrées $|\Delta X|$ et $|\Delta Y|$ du générateur peuvent être soit les registres $|\Delta X|$ et $|\Delta Y|$, soit des bits du registre de commandes dans

le cas de petits vecteurs. Dans ce cas, les 6 bits de poids forts doivent être forcés à 0.

e) Les vecteurs peuvent être tracés en pointillés: 2 bits du registre de contrôle indiquent le "type de trait" (voir§2.11). Ceci peut se réaliser à l'aide d'un compteur 4 bits et d'un multiplexeur 4→1 (Fig.5.22).

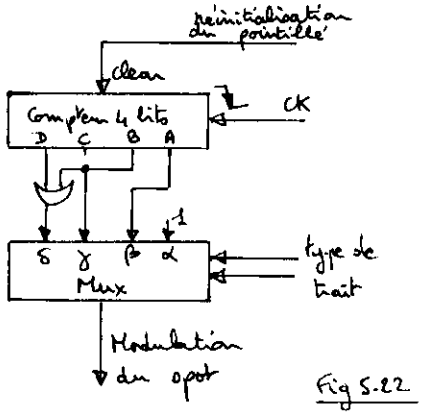
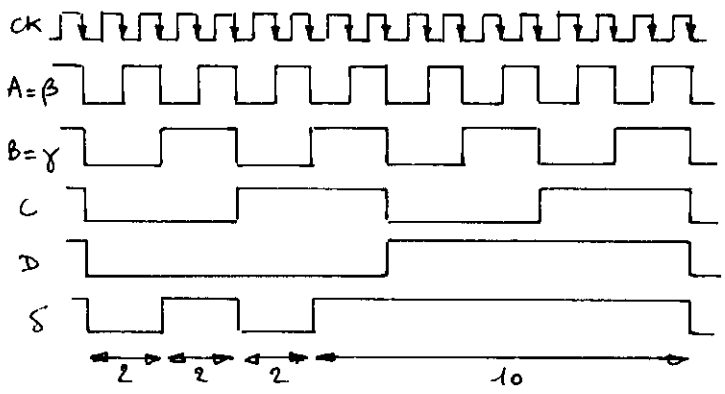


Fig 5.22



La réinitialisation du pointillé peut se faire sur une des phases d'initialisation du vecteur ce qui garantit que les pointillés commencent par la même séquence à chaque début de traçage d'un vecteur.

Mais ce compteur 4 bits de pointillés fait vraiment double emploi avec le compteur UP/DOWN du générateur dont on peut utiliser les 4 sorties de poids faible. Dans ce cas, 2 vecteurs différents ne commenceront pas forcément par la même séquence de pointillé, mais on est sûr qu'un même vecteur commencera toujours par la même séquence, et cela suffit pour garantir qu'une figure donnée puisse dans tous les cas être effacée en la retraçant après avoir uniquement changé la bascule Marquant/Effaçant (§2.11).

f) Comme les compteurs X et Y sont synchrones avec l'horloge CKIN, les sorties du générateur de vecteurs ne doivent pas être CKX,CKY mais EnableX, EnableY (voir Fig.5.23).

g) Tous les éléments du générateur recevant CKIN doivent

recevoir aussi "Enable écriture" pour se bloquer lors des périodes de visualisation. En effet, il ne sert à rien de faire tourner le générateur pendant ces périodes pour anticiper l'écriture car alors cela nécessiterait un tampon de 64 couples d'adresses (X,Y)! (par contre, si on veut augmenter la vitesse de ce générateur et si on utilise des mémoires d'image statiques, on peut mettre à l'état haut l'entrée FECR).

5.13-SCHEMA DE PRINCIPE

Le générateur peut se schématiser comme en figure 2.22 et sa structure interne comme en Fig.5.23. Ce schéma ne demande qu'une petite remarque supplémentaire: il n'est pas nécessaire de mettre Enable écriture sur la bascule D en sortie de l'additionneur B, si ce n'est pour diminuer la consommation électrique du circuit en évitant que le multiplexeur et l'additionneur B ne commutent constamment lors des phases de visualisation.

5.14-MAQUETTE DE SIMULATION MSI

Le plan de celle-ci est donné Fig.5.24 et 5.25. La figure 5.25 représente les 2 registres $|\Delta X|$ et $|\Delta Y|$, leur chargement par le bus 3 états interne et leur lecture sur ce bus.

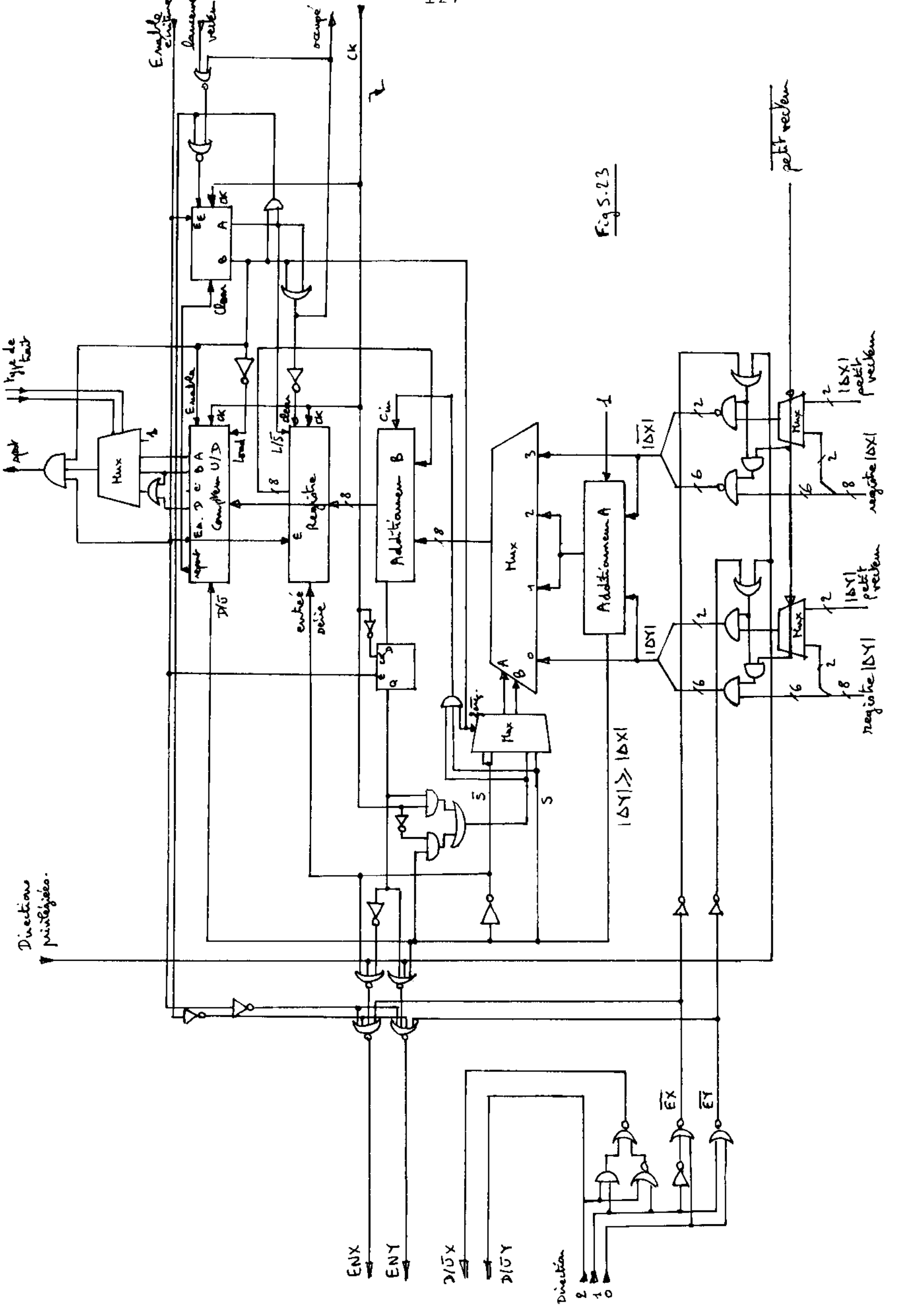

Enable
lancement vecteur
occupé
CK
Clear
Compteur U/D
Mux
Registre
entrée série
Additionneur B
Additionneur A
Mux
|ΔY| ≥ |ΔX|
petit vecteur
registre |ΔX|
registre |ΔY|
Direction privilégiée
ENX
ENY
D/U X
D/U Y
Direction
2
1
0

Fig 5.23

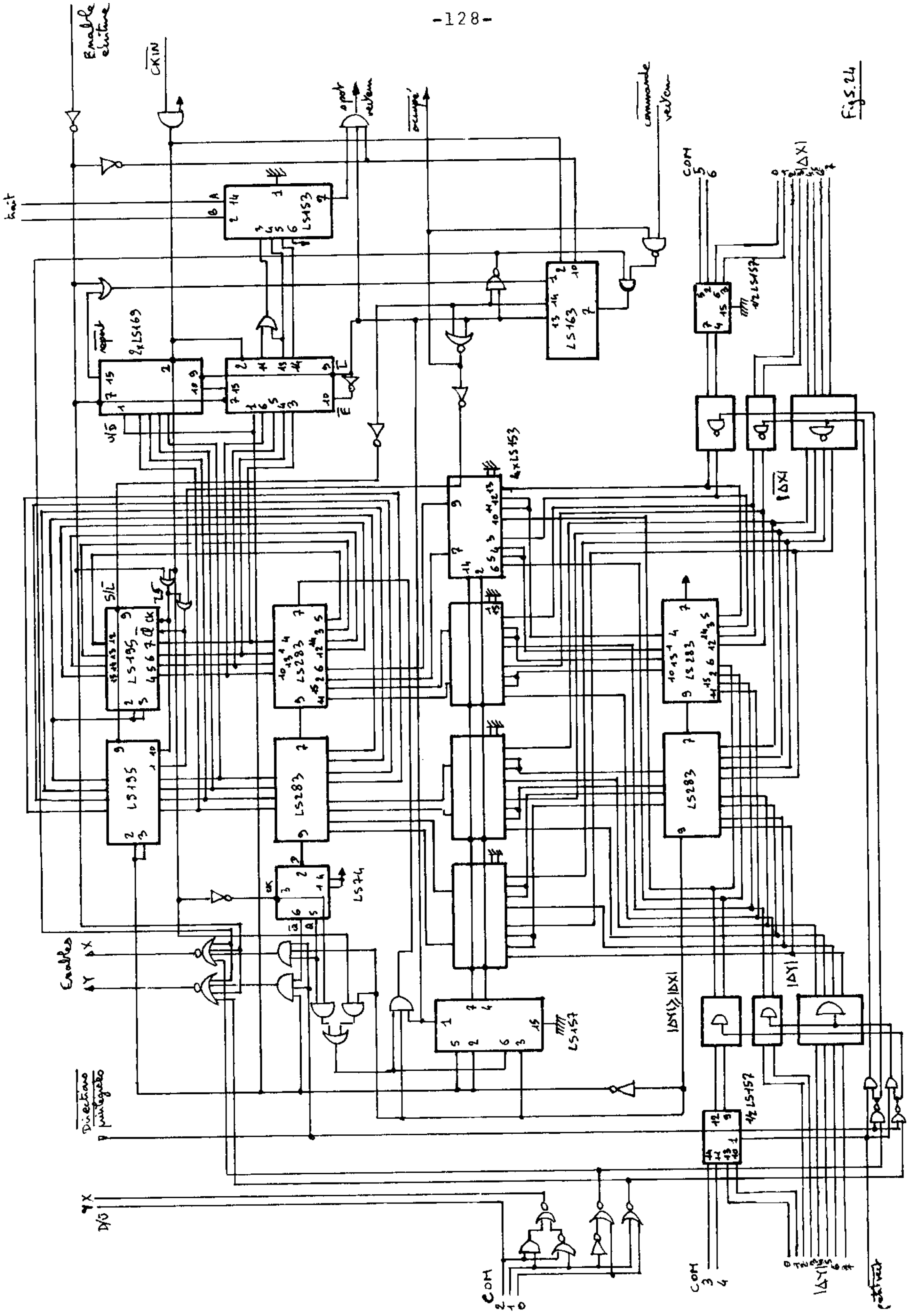

Enable écriture
CKIN
Init
LS153
LS163
2xLS169
LS195
LS283
LS74
LS157
4xLS153
4xLS157
COM
|ΔX|
|ΔY|
|ΔY|≥|ΔX|
Enables
Direction privilégiée

Fig.5.24

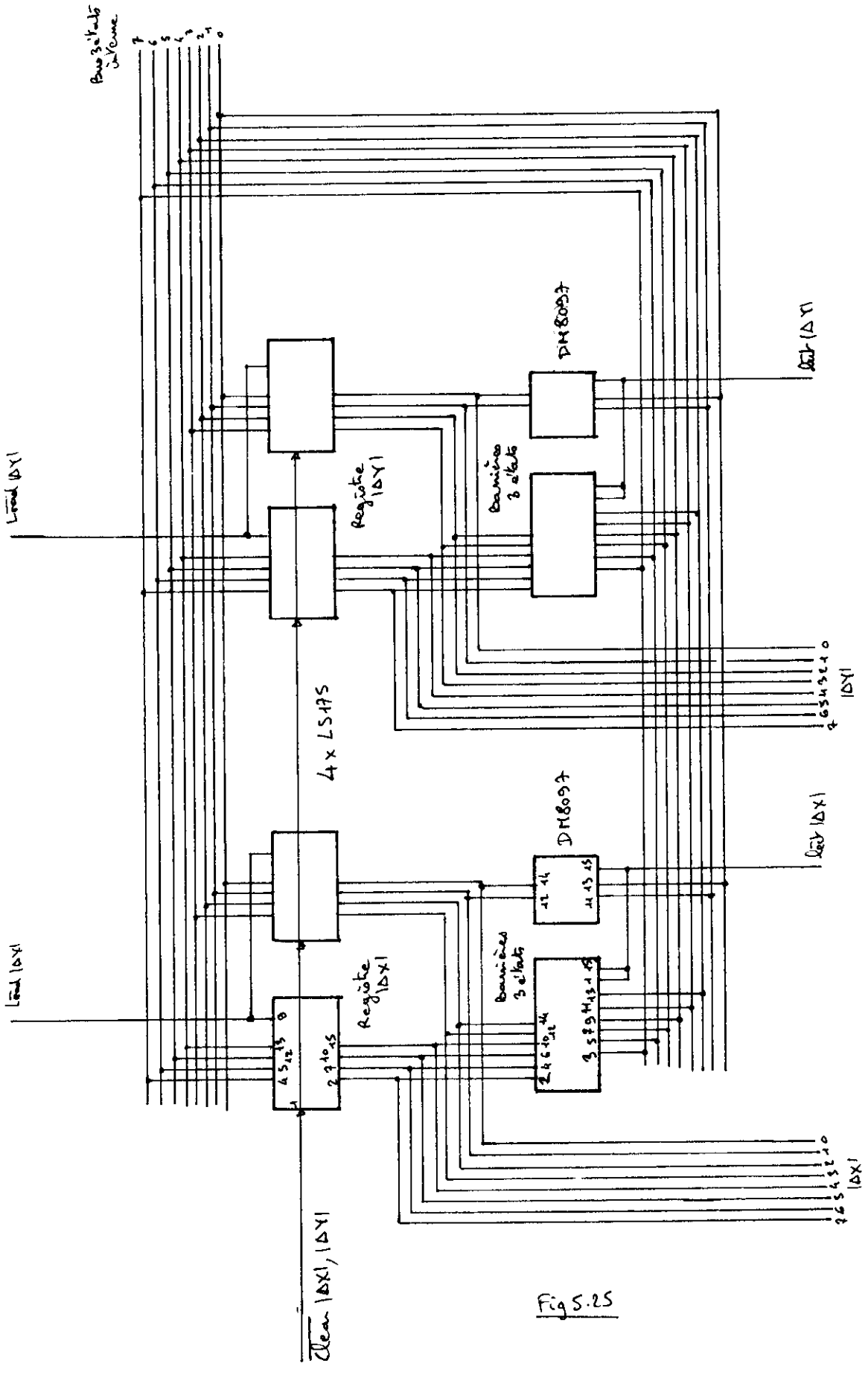

Bus 8 bits interne
Load ΔY1
Registre ΔY1
Bascules 8 bits
DM8097
4 x LS175
Load ΔX1
Registre ΔX1
Bascules 8 bits
DM8097
Clear ΔX1, ΔY1

Fig 5.25

5.15-IMPLANTATION SUR LE CIRCUIT INTEGRE

Ce générateur de vecteurs comprend une succession d'éléments de 8 bits:

-latch $|\Delta X|$

-latch $|\Delta Y|$

-additionneur

-multiplexeur

-additionneur

-registre

-compteur

Nous pouvons donc nous attacher à implanter une "tranche" de 1 bit qui sera répétée 8 fois, puis à rajouter ensuite quelques briques concernant le reste de la logique.

a) Latchs $|\Delta X|$ et $|\Delta Y|$. On utilise 2 briques 3.1 ayant la commande Clear commune et on sort du même côté les inverses $|\overline{\Delta X}|$ et $|\overline{\Delta Y}|$.

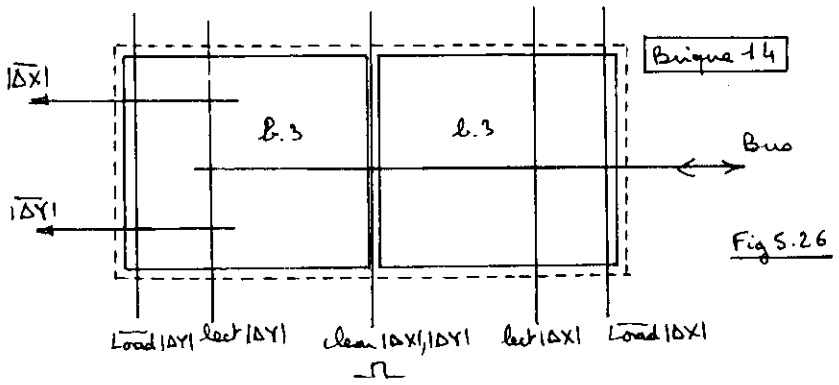


Fig 5.26

b) Multiplexeur à 3 entrées. A l'entrée de ce multiplexeur, on va disposer de $|\Delta Y|$, $|\Delta X|$ et $\overline{|\Delta Y| - |\Delta X|}$ (voir §5.15.i). De plus, on demande une sortie inversée. Il constituera la brique 15.

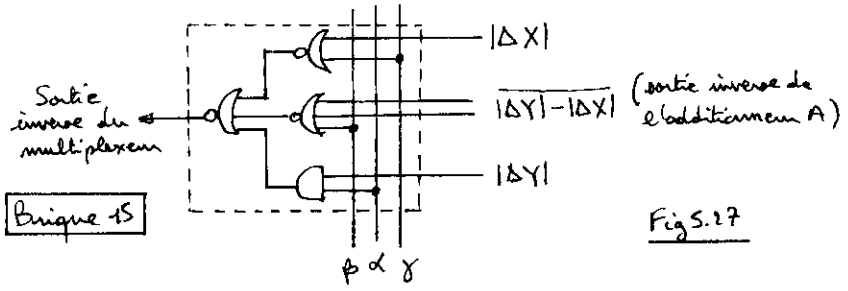


Fig 5.27

La commande de ce multiplexeur est représentée Fig.5.23 par le schéma suivant:

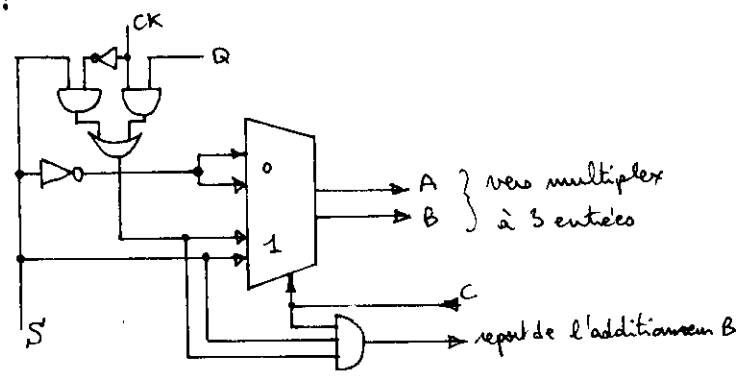


Fig 5.28

Cette schématisation était utile au niveau de l'exposition du fonctionnement, mais au moment de l'implantation, le nombre de portes et de couches logiques peut être abaissé.

Pour commencer, il faut transformer AB en $\alpha\beta\gamma$ suivant la table:

Fig 5.29

A	B	α	β	γ
0	0	1	1	1
0	1	0	0	1
1	0	0	0	1
1	1	0	1	0

Ensuite, α, β, γ, et le report d'entrée de l'additionneur B peuvent se calculer en 2 couches en considérant que:

α n'est à 0 que si
- C=0 et S=0
- ou C=1 et S=1
- ou C=1 et CK=1 et Q=1 (S=0)

β n'est à 0 que si
- C=1 et CK=1 et S=0 et Q=1
- ou C=1 et CK=1 et S=1 et Q=0

γ n'est à 0 que si
- C=0 et S=0
- ou C=1 et CK=0 et S=1
- ou C=1 et CK=1 et S=1 et Q=1

le report d'entrée de l'additionneur B n'est à 0 que si
- C=0 (S=1)
- ou S=0 (C=1)
- ou CK=1 et Q=0 (C=1,S=1)

ce qui conduit à la brique de génération de ces 4 fonctions:

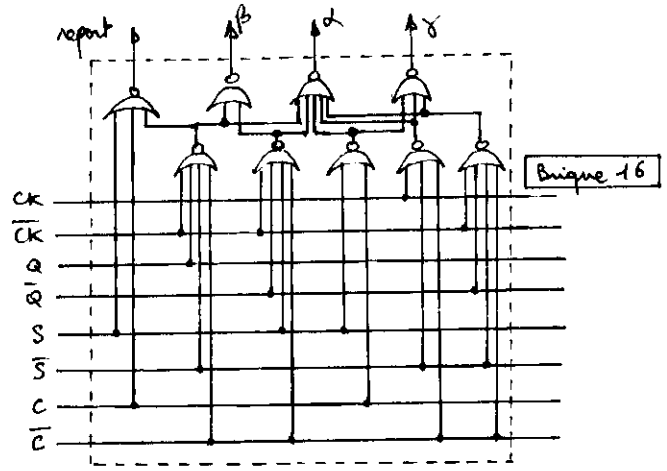


Fig 5.30

c) Registre. Il nous faut un registre à commande Load/Shift, avec Enable, et Clear synchrone. On peut le faire à partir de la brique 4

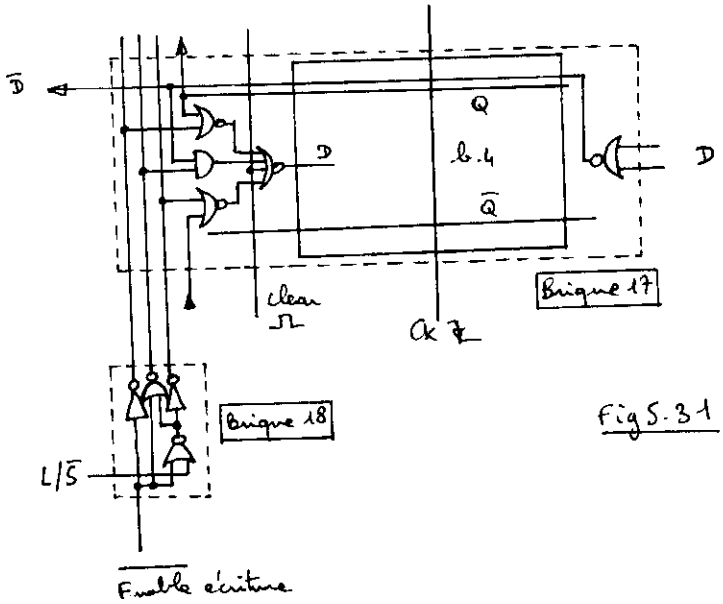


Fig 5.31

Le Clear est prioritaire sur les autres commandes et ne nécessite pas de validation par Enable écriture

d) Compteur UP/DOWN. C'est un compteur UP/DOWN synchrone avec Enable, et Load synchrone. Nous pouvons le construire à partir des briques 4, 9.2, 11.1, 11.2. $\overline{\text{Load}}$ bas doit forcer $\overline{T}$ haut (voir §3.9.i). De plus, indépendamment de la structure de cette brique 9.2, il ne faut pas que le compteur puisse calculer un report de sortie lors du

chargement, sinon il risquerait de repasser directement à la phase 0 (voir Fig.5.20 et 5.23). Par contre, il n'est pas gênant que $\overline{Load}$ soit bas lorsque $\overline{Enable\ écriture}$ est haut puisque le compteur ne compte pas, on se moque de son état qui sera de toutes façons rechargé dès que $\overline{Enable\ écriture}$ reviendra bas.

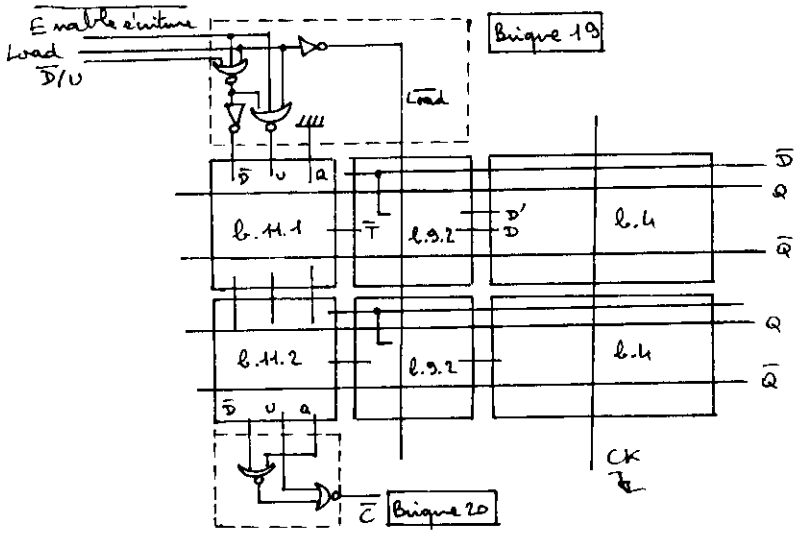


Fig 5.32

e) <u>Forçages sur |ΔX| et |ΔY|.</u> Les sorties des latchs $|\overline{\Delta X}|$ et $|\overline{\Delta Y}|$ doivent pouvoir être forcées pour forcer |ΔX| ou |ΔY| à 0. Les bits faibles doivent en outre pouvoir porter des valeurs de |ΔX|, |ΔY| venant du registre de commande. Ce qui se réalise de la façon suivante:

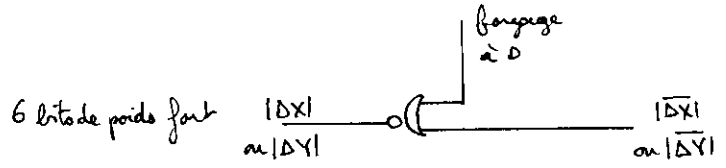
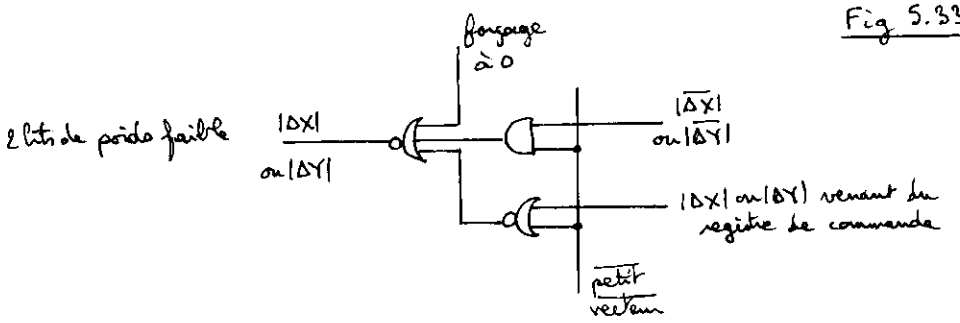


Fig 5.33



Les signaux de forçage peuvent se commander par une logique telle que la suivante:

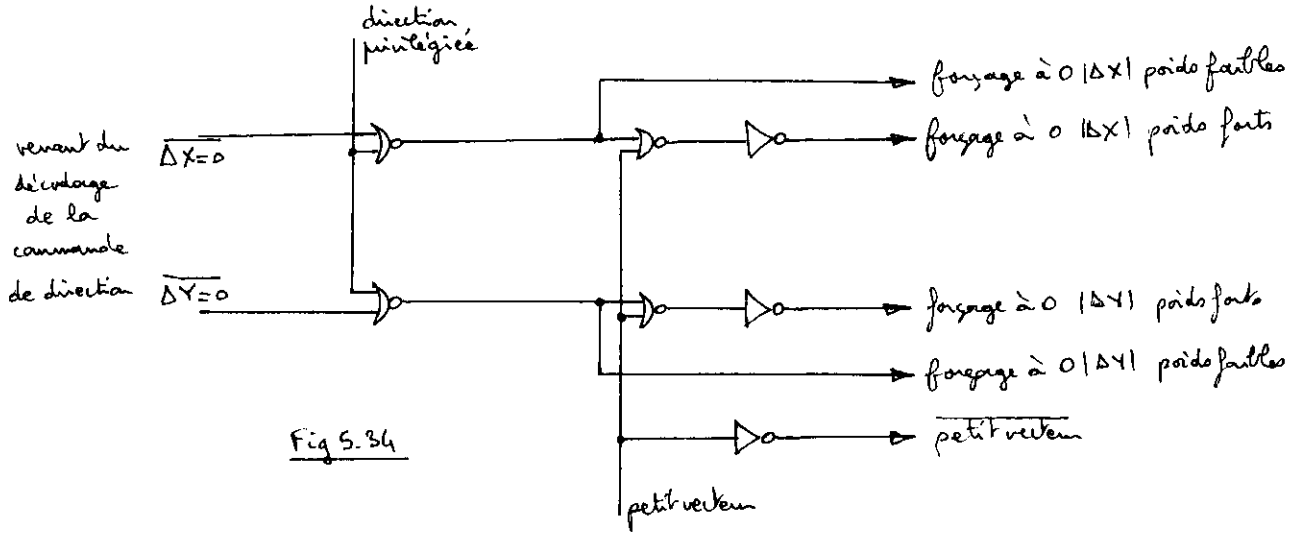


Fig 5.34

f) <u>Décodage de la direction et contrôle des sorties ENX et ENY du générateur.</u> Il faut adjoindre le montage à gauche de la figure 5.23, qui avec les circuits de e) ci-dessus constituent une nouvelle brique:

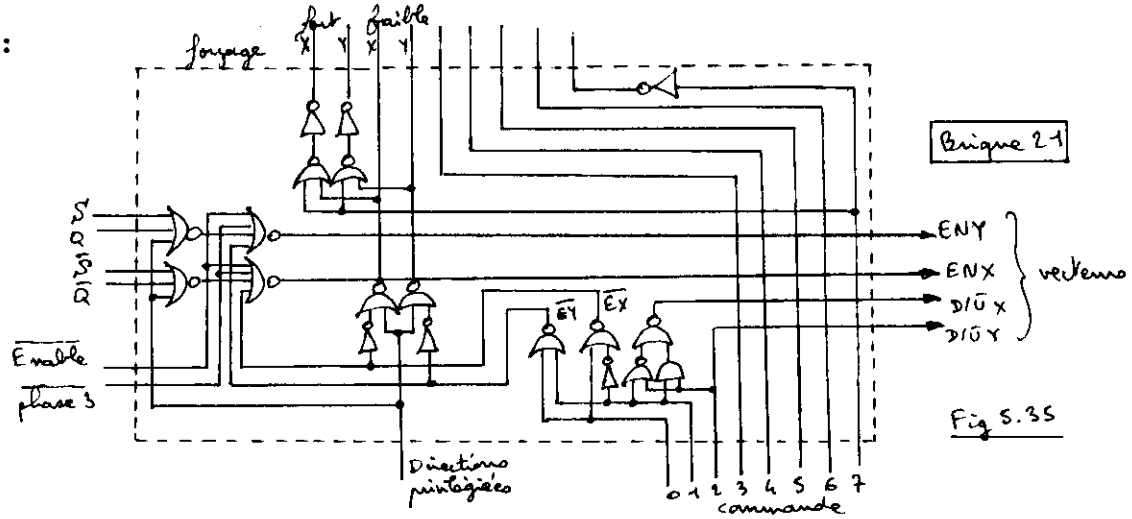
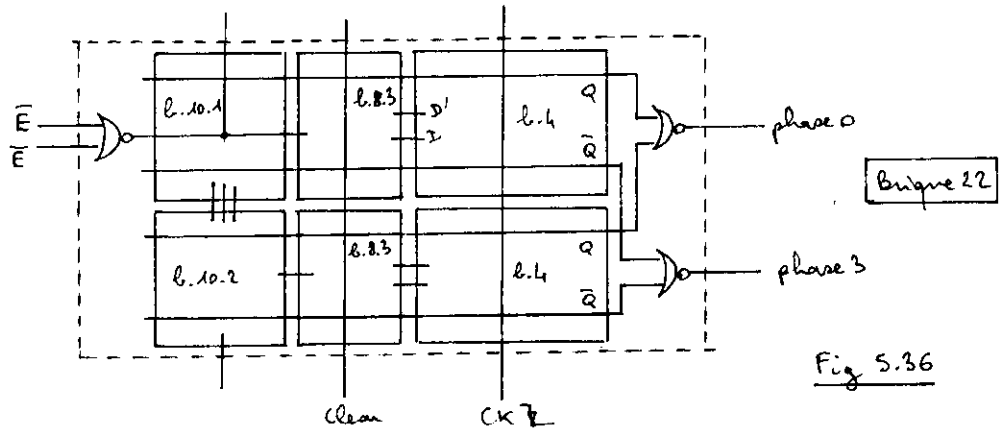


Fig 5.35

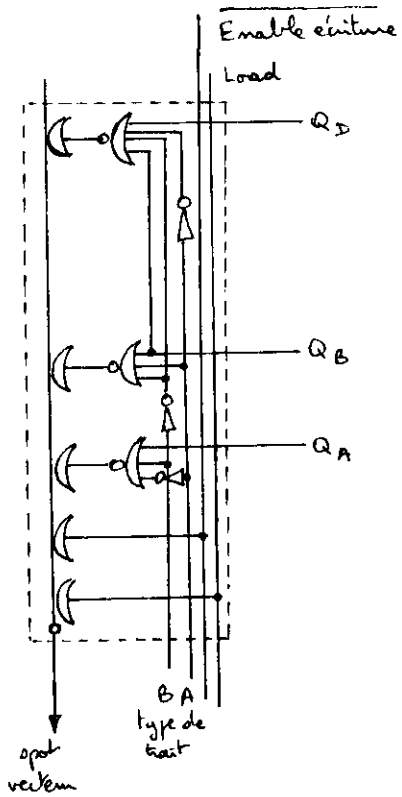
g) <u>Petit compteur 2 bits avec ses reconnaisseurs.</u> C'est un compteur UP synchrone avec Clear synchrone et 2 entrées Enable. Il peut être réalisé à partir des briques 4,8.3,10.1 et 10.2. On a besoin d'un reconnaisseur de l'état 0 et d'un reconnaisseur de l'état 3.



Brique 22

Fig 5.36

h) <u>Génération des pointillés.</u> Le multiplexeur avec un OU sur une entrée et une porte de validation en sortie des Fig.5.22 et 5.23 peut s'implanter de la façon suivante:



Brique 23

Fig 5.37

i) Structure d'une tranche de 2 bits.

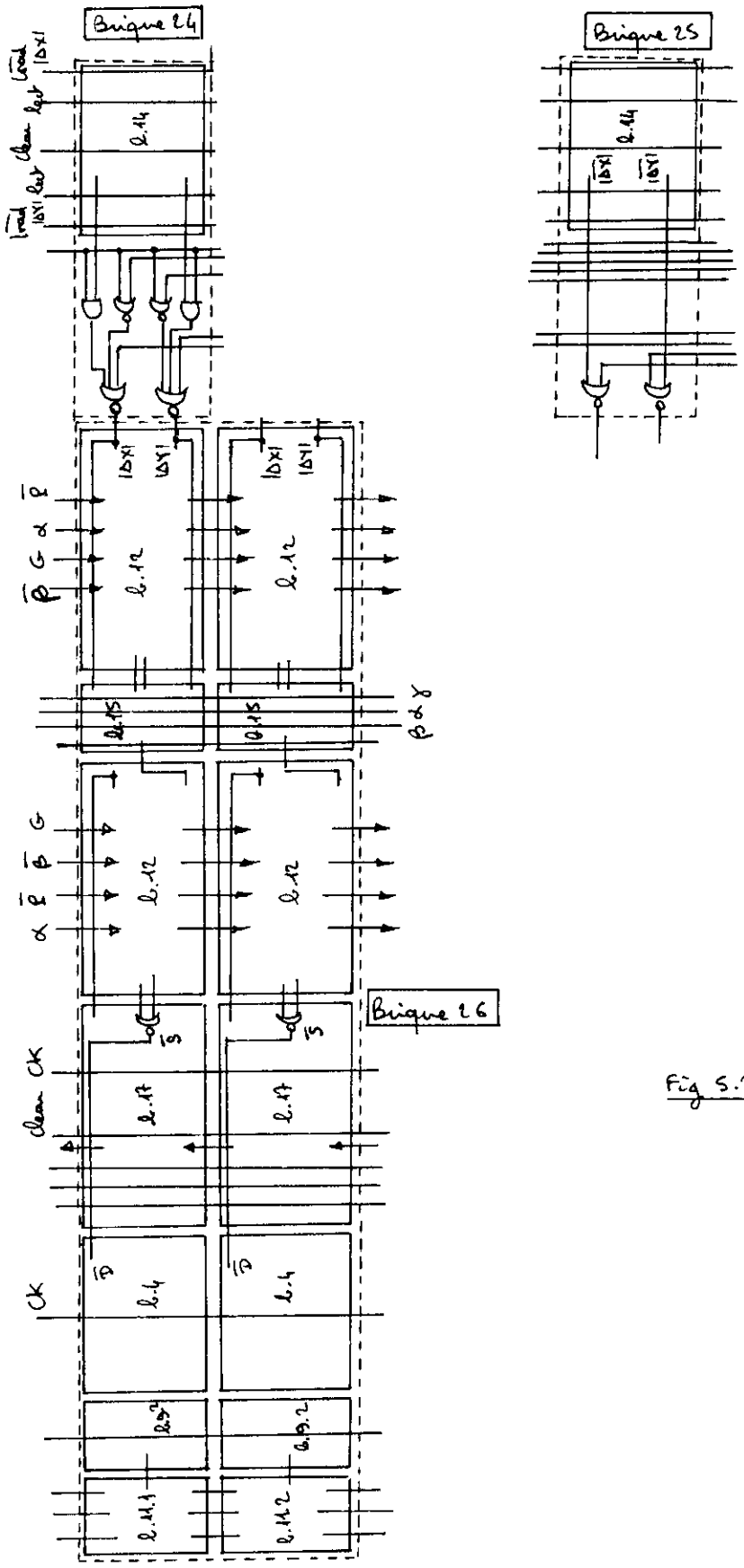
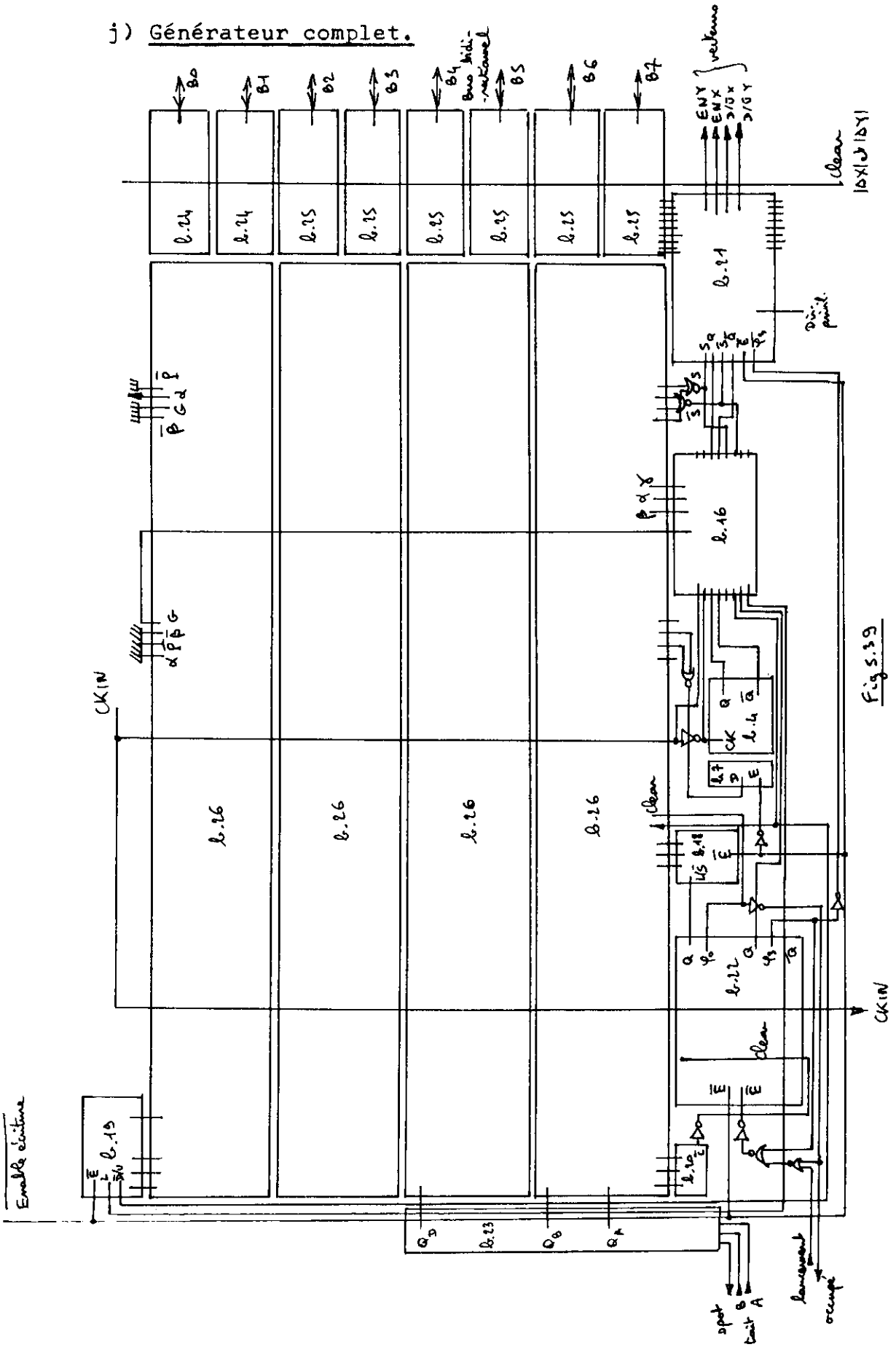


Fig 5.38

j) Générateur complet.



6-GENERATEUR DE CARACTERES

6.1-INTRODUCTION

Nous nous proposons de dessiner un caractère tel que celui de la figure 2.23. Les spécifications externes de ce générateur sont données §2.12 et Fig.2.24. Contrairement au générateur de vecteurs, le chemin suivi par le spot lors du traçage est toujours le même. Le code du caractère peut plutôt se comparer au "type de trait" des vecteurs puisqu'il n'intervient qu'au niveau de la modulation du spot.

6.2-CHOIX DU PARCOURS DU SPOT

Il reste à choisir un moyen pour parcourir la surface des 35pq points du caractère et pour se rendre au point d'arrivée. Ce parcours prendra au moins 35pq+p=(35q+1)p périodes d'horloge. Il faut chercher à approcher ce minimum le mieux possible. Ce qui impose que chaque nouveau point atteind sur un front de l'horloge

- -doit être un des points à rencontrer,
- et -ne doit pas être un des points déjà rencontrés.

Ceci nous suggère un parcours en "zig-zag":

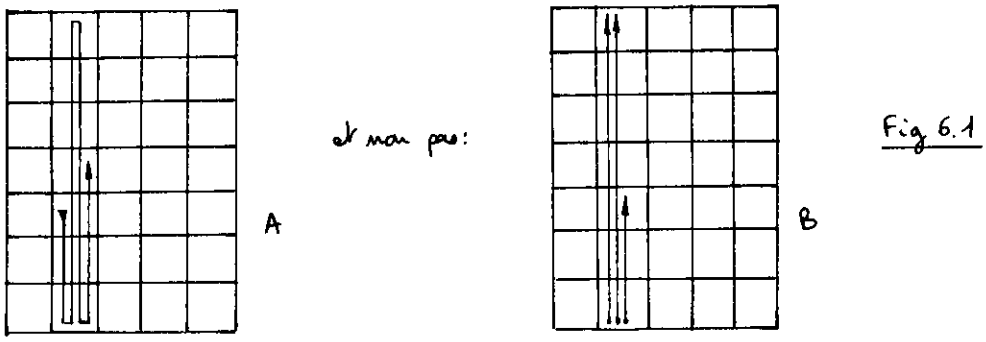


Fig 6.1

En effet, la solution B imposerait pour aller de la fin d'une ligne à la ligne suivante:

-soit de passer par des points déjà vus ou des points à revoir,

-soit de faire un saut de 7q points, ce qui nécessite un additionneur.

La solution A n'a pas cet inconvénient. Elle n'est sûrement pas la seule. On peut imaginer des parcours en spirale par exemple. Mais elle paraît plus simple à réaliser. Citons 2 avantages qui sautent aux yeux:

-UP/DOWN X est toujours UP,

-UP/DOWN Y est la parité de X.

Le générateur de caractères doit s'arrêter en (X_o+6p, Y_o). Cet espace à droite du caractère peut lui aussi être parcouru en zig-zag mais prend alors $p \times 7q$ périodes d'horloge. On peut l'optimiser facilement. Remarquons que suivant la parité de p, il prend dans ce cas 2 chemins différents:

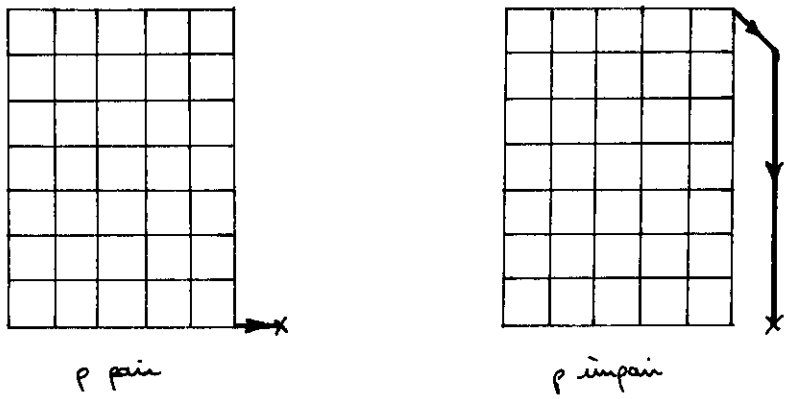


Fig 6.2

Comme les caractères représentent jusqu'à 16x7x16x5 points, il faut un registre d'état de notre automate de 14 bits que l'on peut organiser en 4+3+4+3 (respectivement). Ce registre sera un compteur découpé en 2 parties: X et Y, chacune découpée elle-même en 2 parties:

-une partie haute adressant la mémoire morte générateur de caractère (ROM),

-une partie basse réalisant le facteur d'échelle (comptant modulo p ou modulo q)

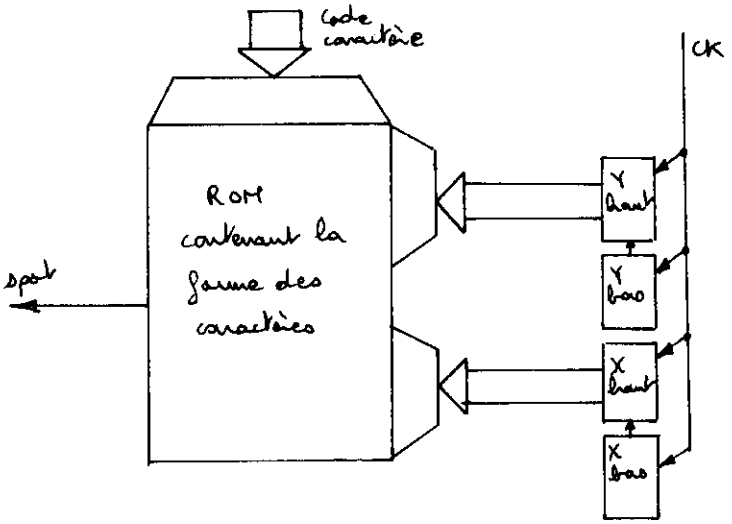


Fig 6.3

Nous indiquerons les états de ces compteurs X ou Y par (μ,ν),

$$\begin{cases} \mu = \text{valeur de la partie haute} \\ \nu = \text{valeur de la partie basse.} \end{cases}$$

6.3-CONVENTIONS D'ADRESSAGE

Les parties hautes des compteurs évolueront de la façon suivante:

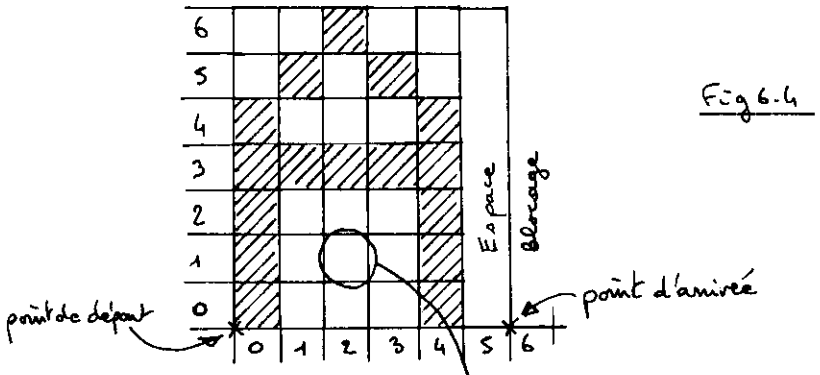


Fig 6.4

pendant que les parties basses évoluent ainsi:

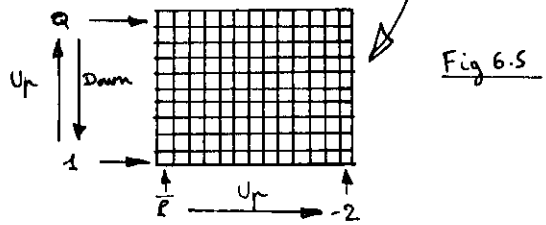


Fig 6.5

Les valeurs extrêmes sont choisies de telle sorte que:

-les compteurs haut et bas d'une même direction comptent dans le même sens (UP ou DOWN)

-les valeurs à charger ou à reconnaître se déduisent facilement de p et q.

6.4-REALISATION DU COMPTEUR X

C'est un compteur UP. La partie haute compte de 0 à 6, la partie basse de $\overline{p}$ à -2. Un reconnaisseur de la valeur -2 sur la partie basse réalise un Load synchrone à $\overline{p}$ de celle-ci, en même temps qu'elle permet l'incrémentation de la partie haute. Le traçage d'un caractère commence par un Clear de la partie haute et un chargement à $\overline{p}$ de la partie basse. L'état 6 de la partie haute bloque tout ce compteur X.

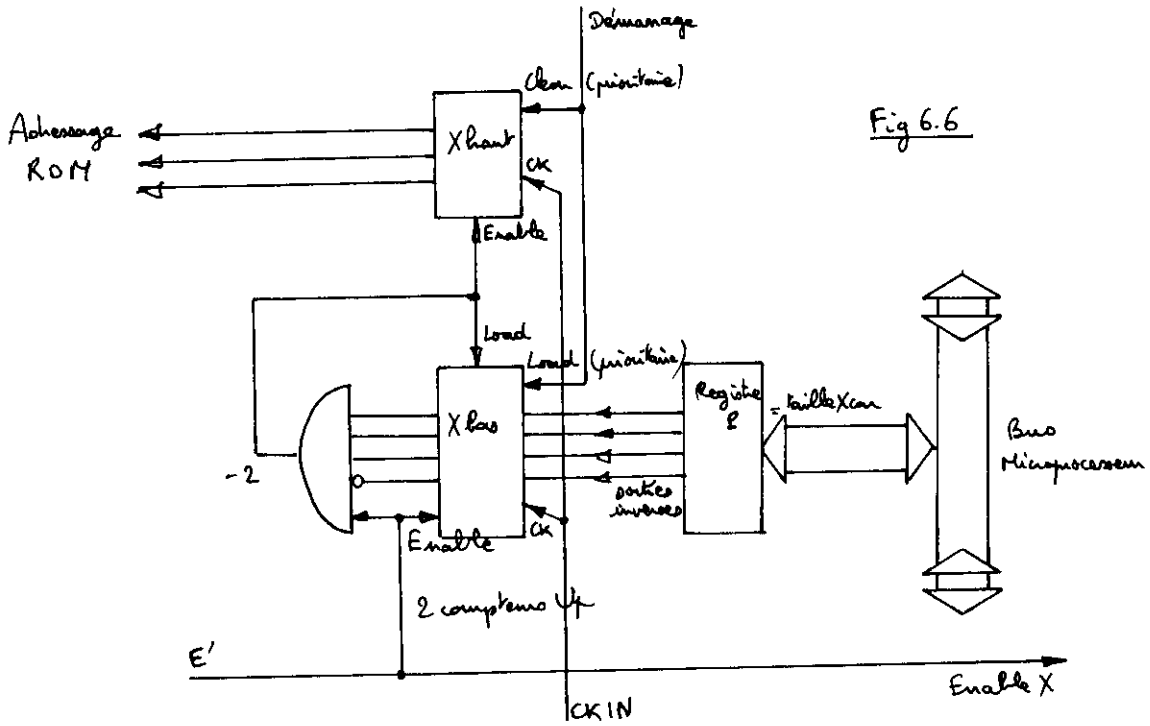


Fig 6.6

Le Enable E' valide l'ensemble de ces 2 compteurs. Ce signal est haut lorsque le spot est bloqué en haut et en bas du caractère à tracer. On a donc:

$$E' = \left\{ (X_{haut} \leqslant 4) \wedge \left[((Y=0,1) \wedge \overline{Xpair}) \vee ((Y=6,q) \wedge Xpair) \right] \right\} \vee (X_{haut} = 5)$$

Ce signal est aussi la sortie EnableX du générateur qui commande le compteur X d'adressage de la mémoire d'image.

6.5-REALISATION DU COMPTEUR Y

Le fonctionnement est analogue, mais les compteurs sont UP/DOWN. Le reconnaisseur sur Y bas est de la valeur 1 ou q (suivant le sens du comptage), et le chargement est respectivement de q ou de 1.

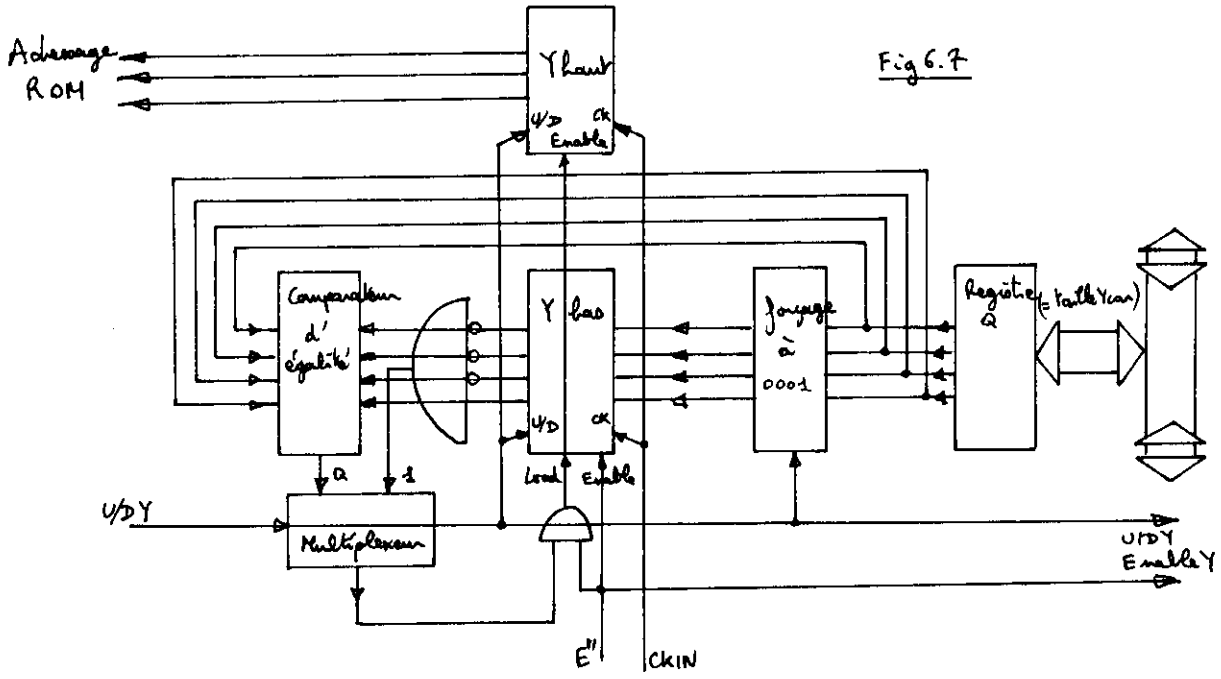


Fig 6.7

Le signal de validation E" est donné par:

$$E'' = \left[(Y \neq (0,1)) \wedge Ydown \right] \vee \left[(Y \neq (6,q)) \wedge Yup \right]$$

Si l'on définit Yup et Ydown par:

$$\begin{cases} Yup = (X \text{ pair}) \wedge (X \text{ haut} \leqslant 4) \\ Ydown = \overline{Yup} \end{cases}$$

alors, l'arrêt du compteur Y ne peut se produire que pour Y=(0,1) et X haut >4. Le compteur Y est donc dans le bon état pour repartir pour un nouveau caractère. L'état de blocage du générateur de caractères est Y=(0,1) et $X \geqslant (6,\bar{p})$ (En général Xarrêt=$(6,\bar{p})$ sauf peut être à la mise sous tension). Cet état est tel que le générateur ne peut démarrer par simple changement de p ou q. Le signal "générateur de caractères occupé" peut être généré par $(X \leqslant 5) \vee (Y \neq (0,1))$

6.6-COMPLEMENTS, ADAPTATIONS

-On a parlé de "parité de X" pour calculer E' et Yup. En fait, il s'agit de la parité du nombre de colonnes entières déjà tracées, ce qui diffère du bit faible du compteur X, puisque si p est impair, on passe de -2 à $\bar{p}$ qui sont 2 valeurs paires. Il faut donc rajouter une bascule T, s'inversant à chaque fois que E' est haut, pour mesurer cette parité.

-Si le temps d'accès de la ROM par rapport aux adresses X,Y est trop long, il faut utiliser sa sortie dans la période qui suit celle de son adressage. Ce qui signifie que toutes les sorties du générateur doivent être tamponnées: UP/DOWN X, ENX, ENY, générateur occupé. (Ce dernier signal ne doit pas être simplement tamponné car il doit être haut dès l'arrivée de la commande de traçage).

-Un état "pavé" doit forcer à "tout allumé" la sortie de la ROM.

-Un état "pavé 4x4" doit forcer le signal Y=(6,q) lorsque Y est (3,q), et doit forcer un Preset du compteur X haut à la valeur 6 lorsque ($(X=(3,-2)) \wedge E'$).

-Le générateur de caractères, tout comme le générateur de vecteurs doit se mettre en "pause" lorsque Enable écriture est bas.

6.7-SCHEMA DE PRINCIPE

Les fonctions E', E", Yup, "générateur occupé", doivent se calculer à partir des reconnaisseurs suivants:

X haut $\leqslant 4$	: α
X haut=5	: β
Y=(0,1)	: γ
Y=(6,q)	: δ
parité de X	: ε

et de la condition Enable écriture (ρ).

$$E' = \rho\beta + \rho\{\alpha . [(\gamma . \bar{\varepsilon}) + (\delta . \varepsilon)]\}$$

$$\boxed{E' = (\beta + \alpha\gamma\bar{\varepsilon} + \alpha\delta\varepsilon)\rho}$$

$$E" = \rho\bar{\gamma} . (\overline{\varepsilon . \alpha}) + \rho\bar{\delta}(\varepsilon . \alpha)$$

$$\boxed{E" = (\bar{\gamma}\bar{\varepsilon} + \bar{\gamma}\bar{\alpha} + \bar{\delta}\varepsilon\alpha)\rho}$$

$$\boxed{Yup = \varepsilon\alpha}$$

$$\boxed{\text{"générateur occupé"} = \alpha + \beta + \bar{\gamma} = \overline{\bar{\alpha}\bar{\beta}\gamma}}$$

ce qui permet de dessiner le schéma de principe de la figure 6.8, dans lequel la ROM est supposée suffisamment rapide.

6.8-SIMULATION MSI

Le schéma de simulation est donné Fig.6.9. Les fonctions combinatoires sont réalisées à partir de mémoires mortes de 256 mots de 4 bits (SFC 71301).

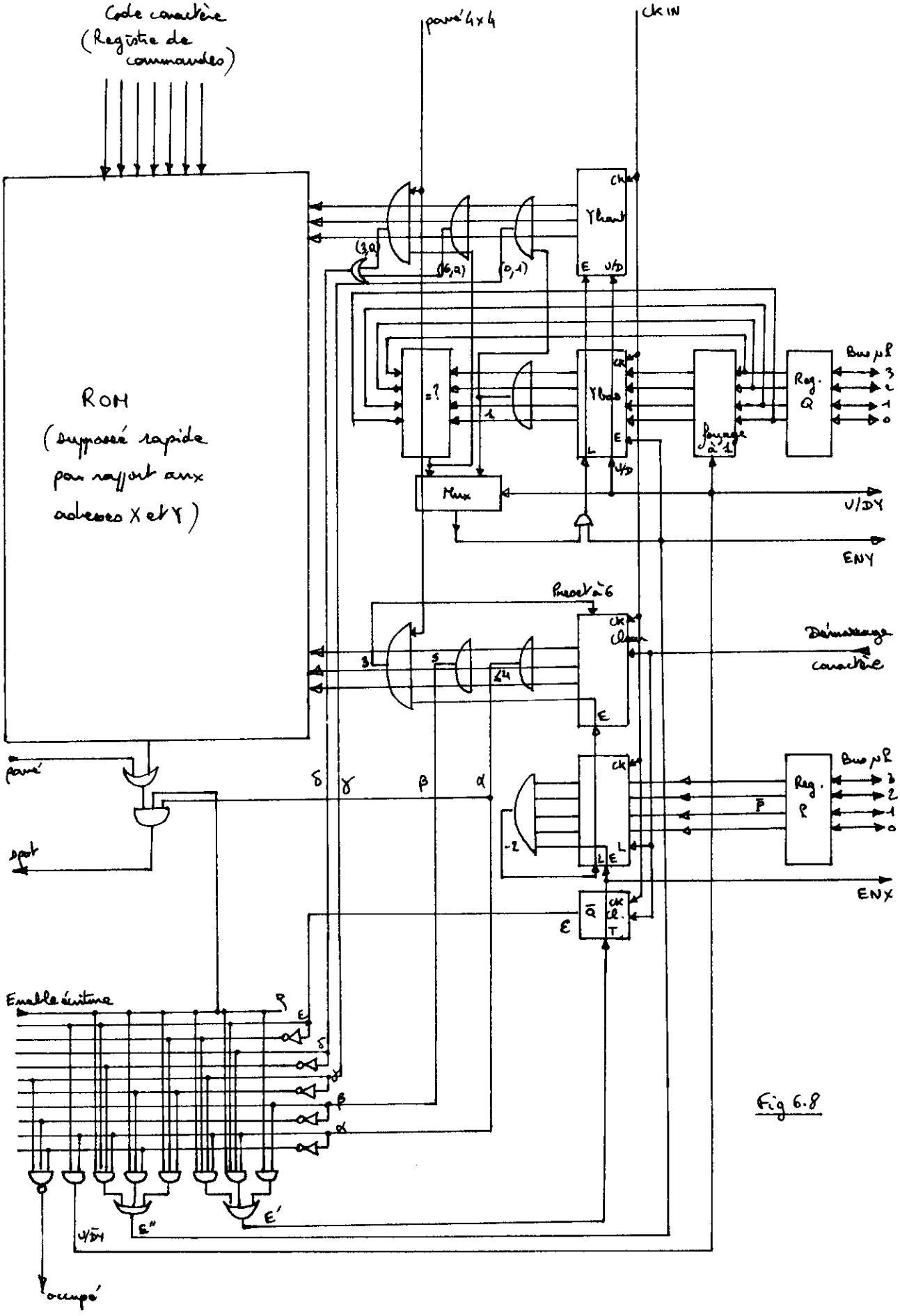

Code caractère
(Registre de commandes)
ROM
(suppose rapide par rapport aux adresses X et Y)
pavé 4x4
CK IN
Ybase
Ybase
Mux
Reg. Q
Bus µP
U/DY
ENY
Passe à 6
Démarrage caractère
Reg. P
ENX
pavé
spot
Enable écriture
occupé

Fig 6.8

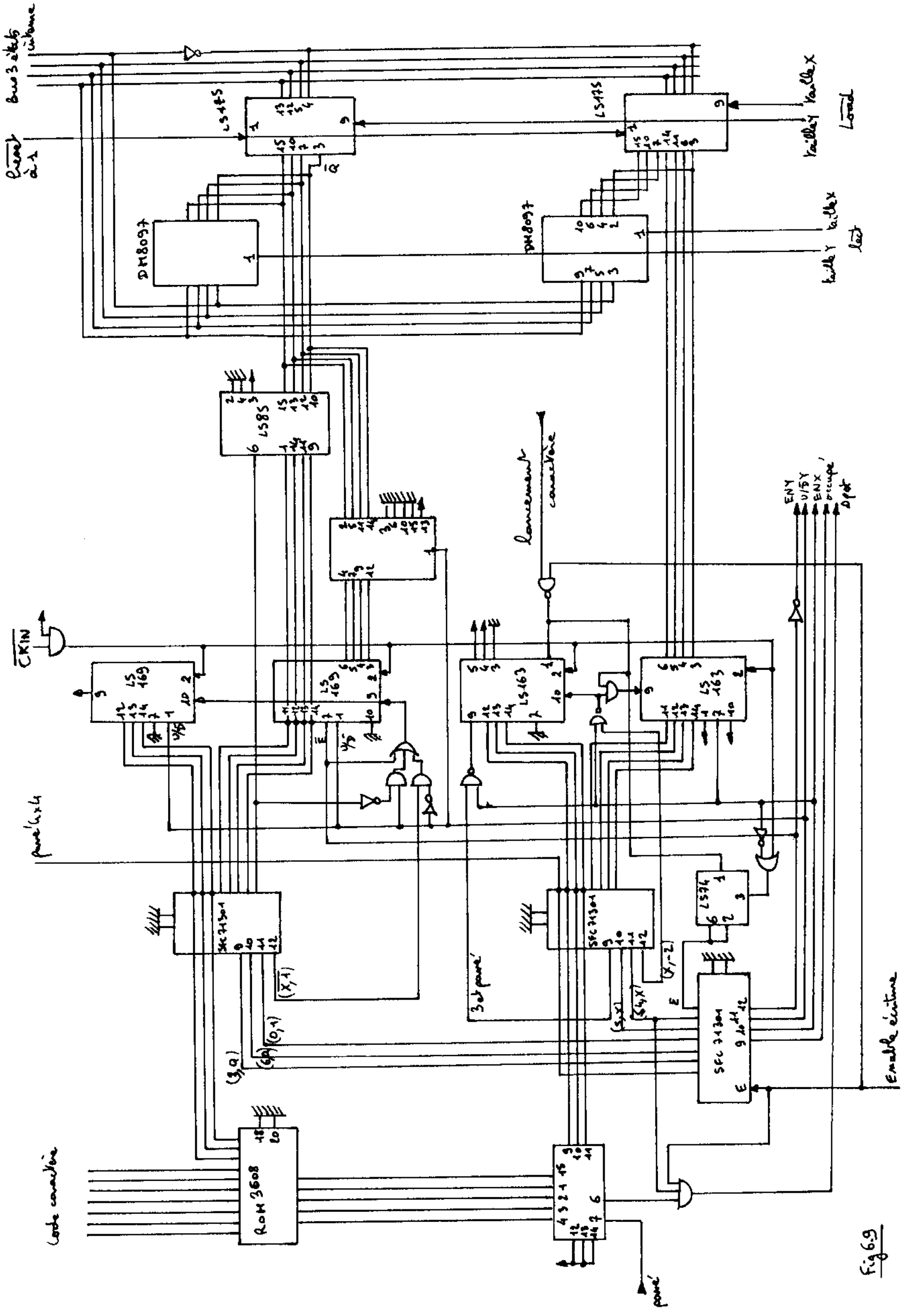

Bus 3 états interne
LS485
LS175
DM8097
DM8097
LS85
LS169
LS169
LS163
LS163
CKIN
SFC 71301
SFC 71301
SFC 71301
LS74
ROM 3608
Load
lancement caractère
Enable écriture
Code caractère
ENY
ENX
Dpt

Fig.6.9

6.9-IMPLANTATION

a) Réalisation du compteur Y haut. C'est un compteur UP/DOWN 3 bits avec Enable. Il peut être réalisé à partir de briques 4, 8.2, 11.1, et 11.2 (Fig.6.10).

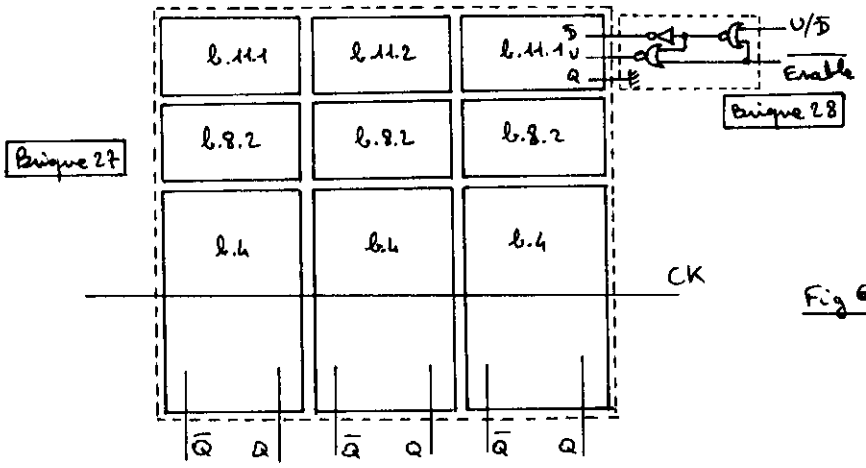


Fig 6.10

b) Réalisation du compteur X haut. C'est un compteur UP 3 bits avec Enable, Clear synchrone et Preset à 6 synchrone. Il peut être réalisé à partir de briques 4, 8.7, 10.3 et 10.4 (Fig.6.11).

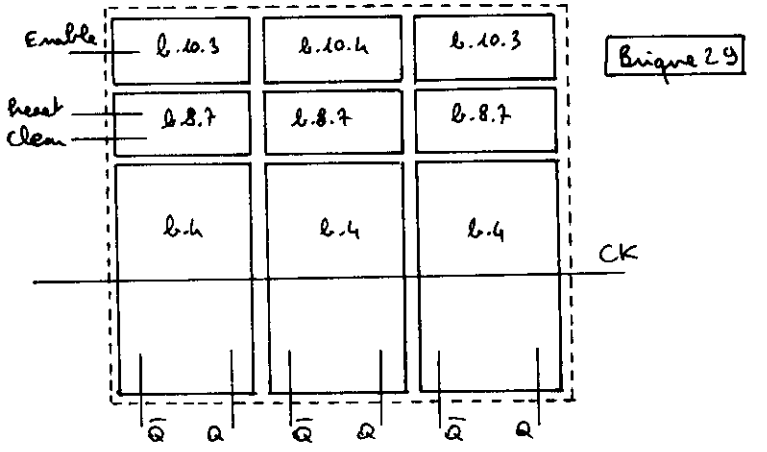


Fig 6.11

c) Réalisation du compteur Y bas. C'est un compteur UP/DOWN 4 bits synchrone, avec Enable et Load synchrone. Il peut être réalisé à partir de briques 4, 9.2, 11.1, 11.2 et 19 (Fig.6.12).

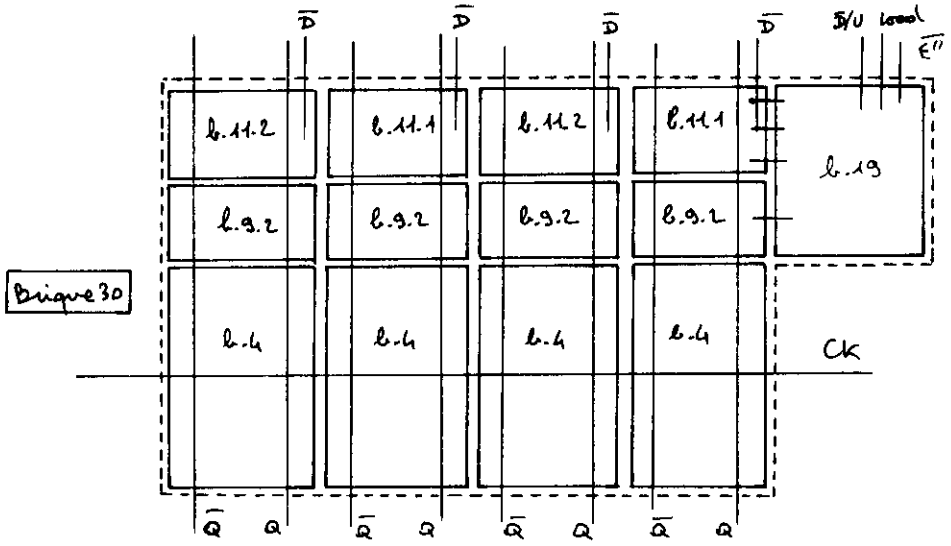


Fig 6.12

d) Compteur X bas et bascule de parité. C'est un compteur UP avec Load et Enable. Il peut être réalisé à partir de briques 4, 9.1, 8.5, 10.1 et 10.2 (Fig.6.13).

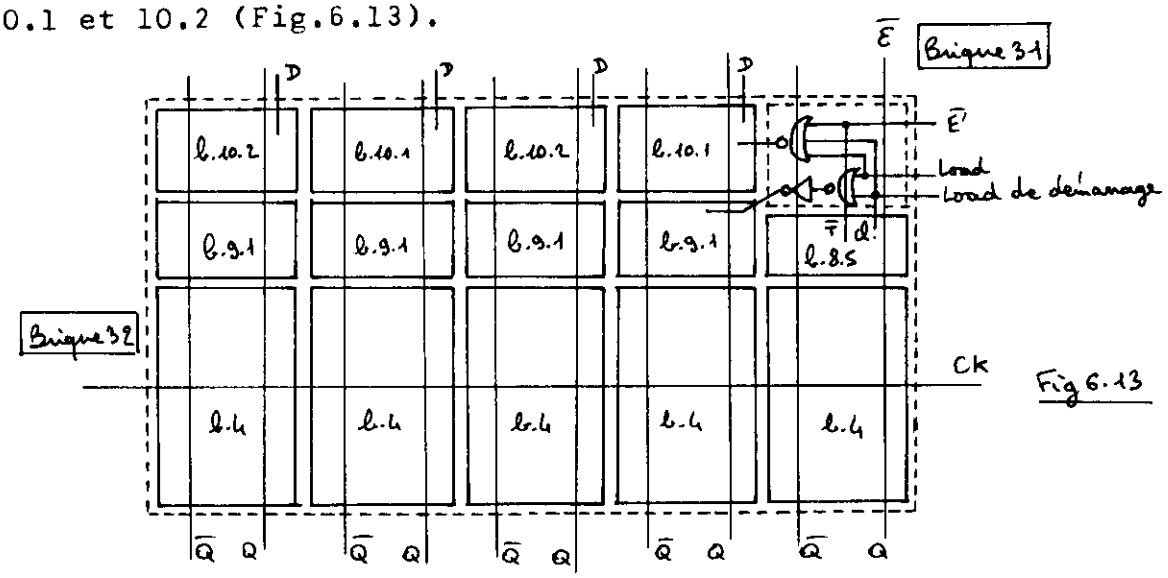


Fig 6.13

e) Latchs tailleX et tailleY. Il s'agit de briques 3.1 et 3.2.

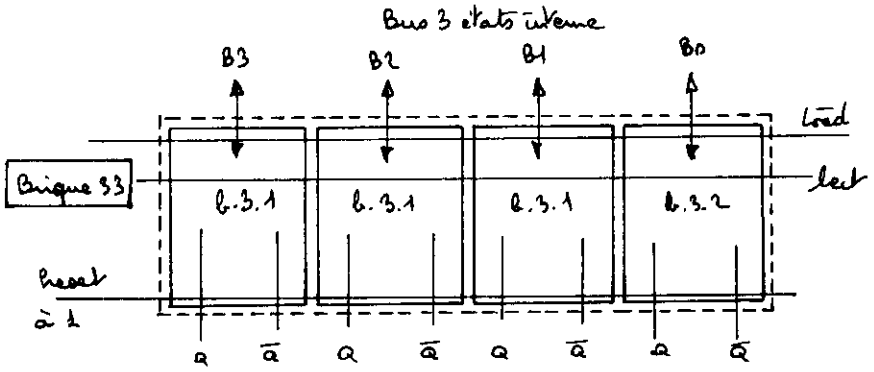


Fig 6.14

f) Forçage à 1 et comparateur d'égalité. Le comparateur se réalise à partir de briques 13.

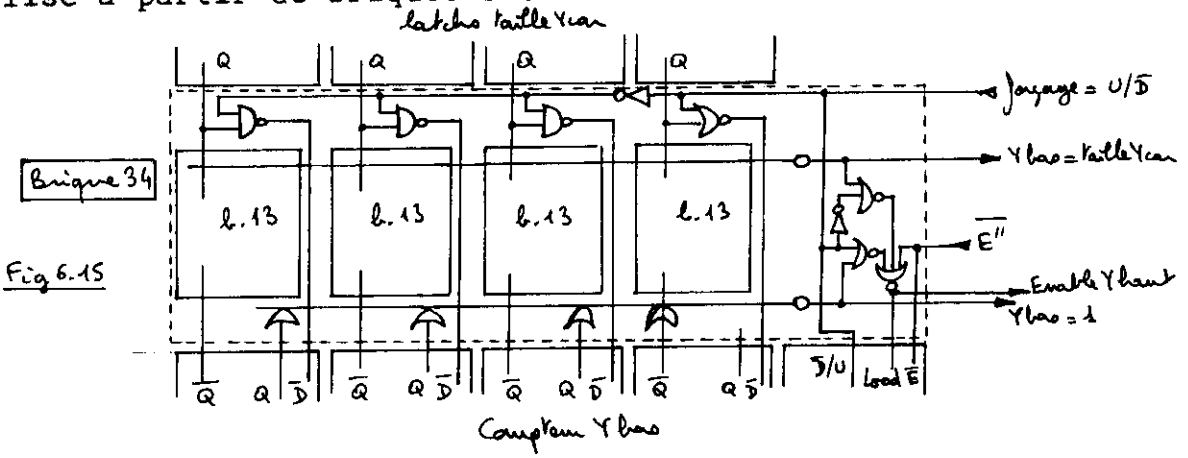


Fig 6.15

g) Fonctions combinatoires. Elles sont au nombre de 5: $\overline{E'}$, $\overline{E''}$, UP/$\overline{\text{DOWN}}$ Y, "occupé" et $\overline{\text{Enable ROM}}$ (voir paragraphe suivant).

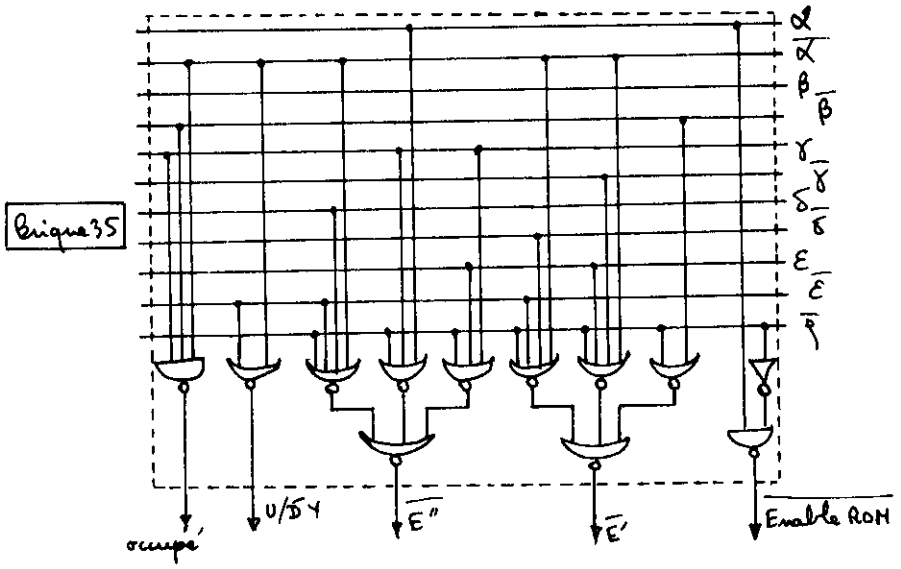


Fig 6.16

h) Mémoire morte. L'adresse de cette mémoire comprend 2 champs:

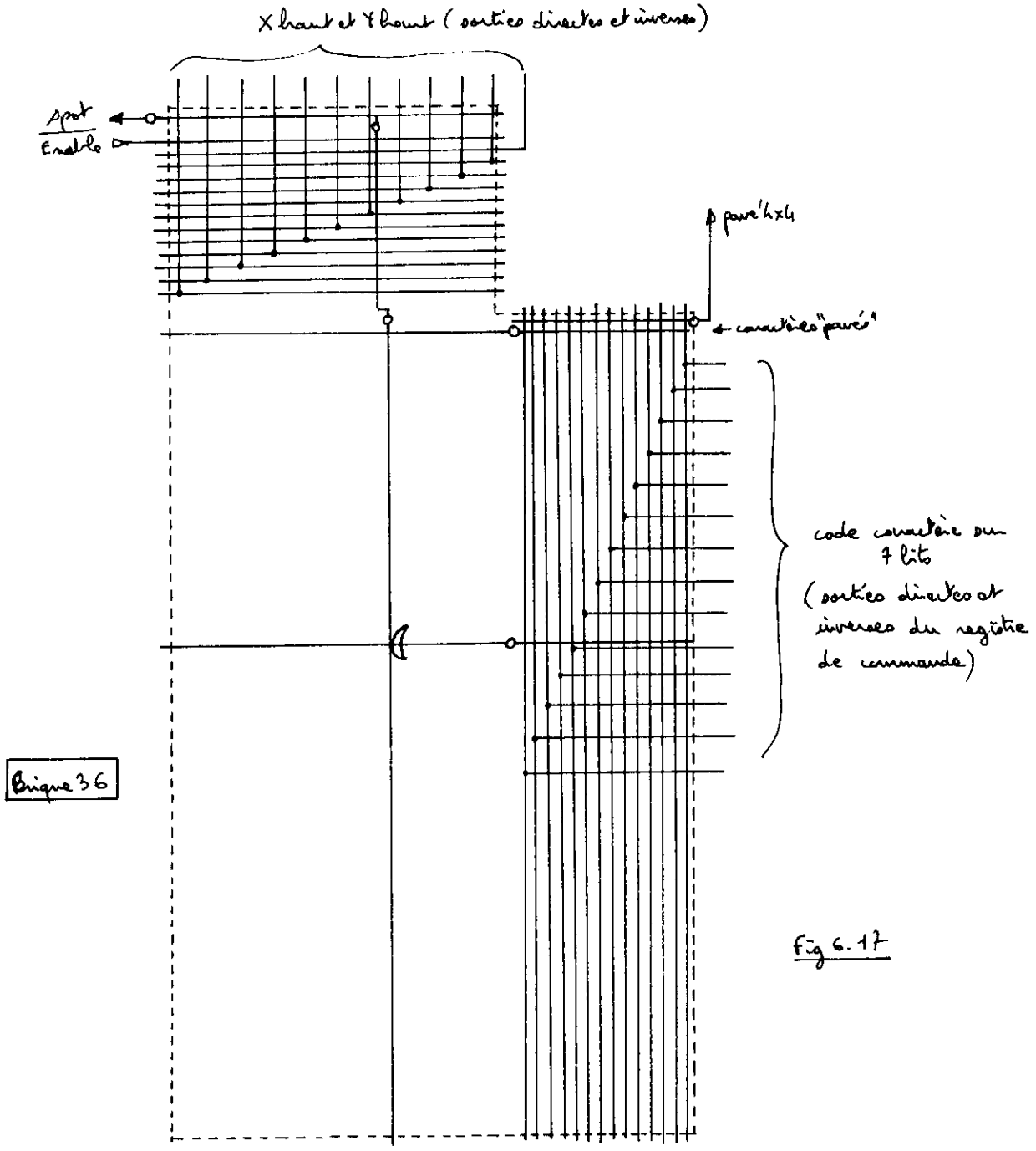
-d'une part le code du caractère (7 bits),

-d'autre part les coordonnées du point dans le caractère (3 bits d'adresse ligne et 3 bits d'adresse colonne).

La mémoire doit être plus rapide pour la deuxième partie de l'adresse que pour la première (code caractère). Ce qui suggère l'implantation de la figure 6.17 (voir aussi §3.8). La présence d'un transistor sur la matrice correspond à "point allumé" sur le caractère (comme il y en a moins, cela diminue la capacité et accélère la ROM). Dans ces

conditions, la sortie de la mémoire est inverse, et la commande est $\overline{\text{Enable}}$.

Cette mémoire comprend 96x35=3360 bits. Un bit peut être implanté sur une surface inférieure à $20 \times 20\ \mu^2$. Les dimensions de la matrice sont donc de l'ordre de $1,9 \times 0,7\ mm^2$.



Brique 36

Fig 6.17

i) Générateur complet. Il s'obtiend en rassemblant les briques 27, 28, 29, 30, 32, 33, 34, 35, 36, et en ajoutant les quelques reconnaisseurs sur les compteurs.

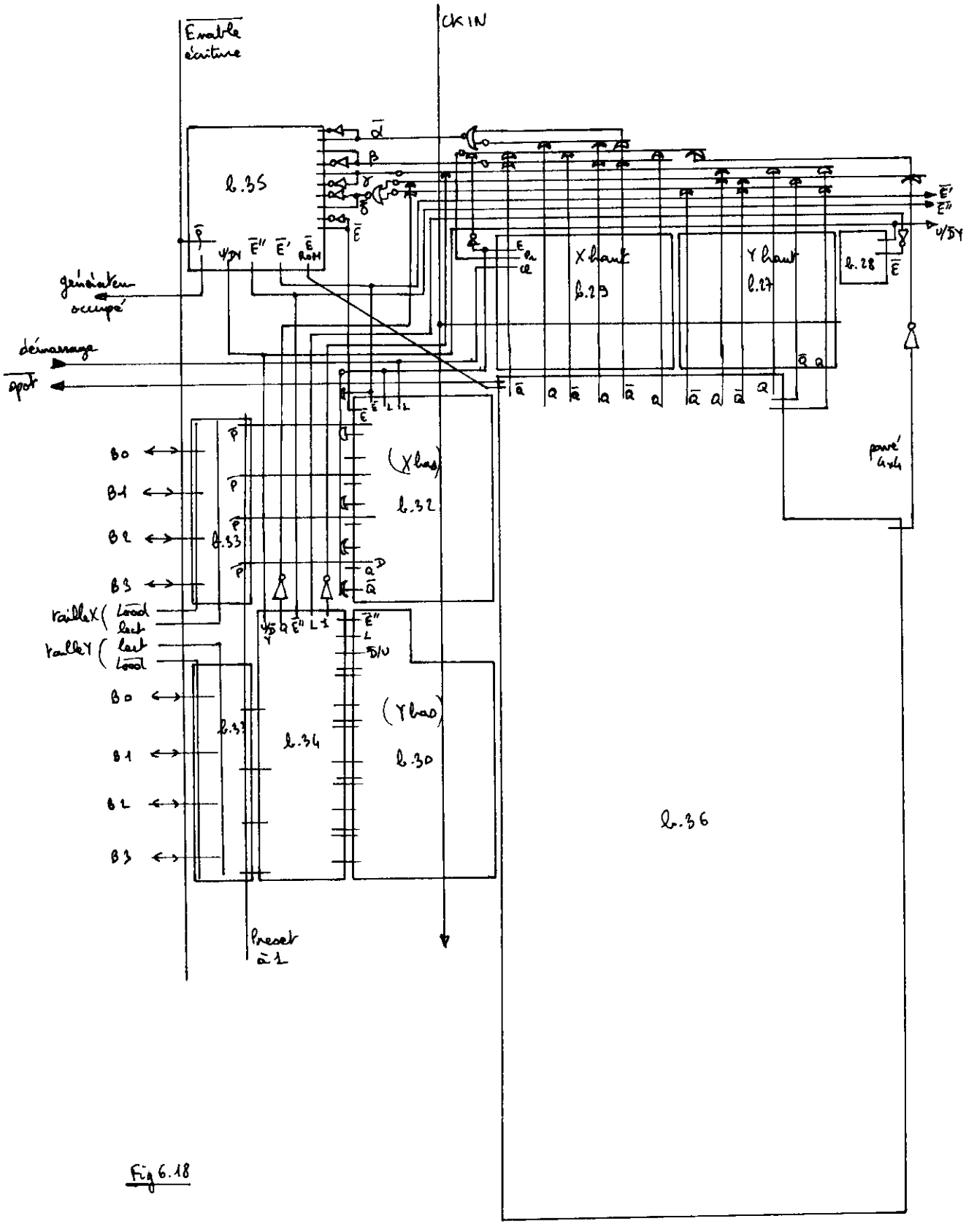


Fig 6.18

7-LOGIQUE DE CONTROLE ET DISPOSITION DES SOUS-ENSEMBLES SUR LA PUCE

7.1-Compteurs X et Y adresses du spot.

7.2-Registre de contrôle, mot d'état et système d'interruption.

a)Registre de contrôle.

b)Signal "libre/occupé".

c)Mécanisme de contrôle des interruptions et mot d'état.

d)Simulation MSI et implantation.

7.3-Logique du light-pen.

7.4-Effacement de l'image ou positionnement d'un fond.

7.5-Registre de commandes, décodage et synchronisation.

7.6-Bus 3 états interne et échanges avec le microprocesseur.

7.7-Disposition des sous-ensembles sur la puce.

7.1-COMPTEURS X ET Y ADRESSES DU SPOT

Ce sont 2 compteurs UP/DOWN synchrones de 12 bits. Ils ont chacun une commande Enable et une commande UP/DOWN issues des générateurs de vecteurs et de caractères, ainsi qu'un Clear synchrone issu du décodage des commandes.

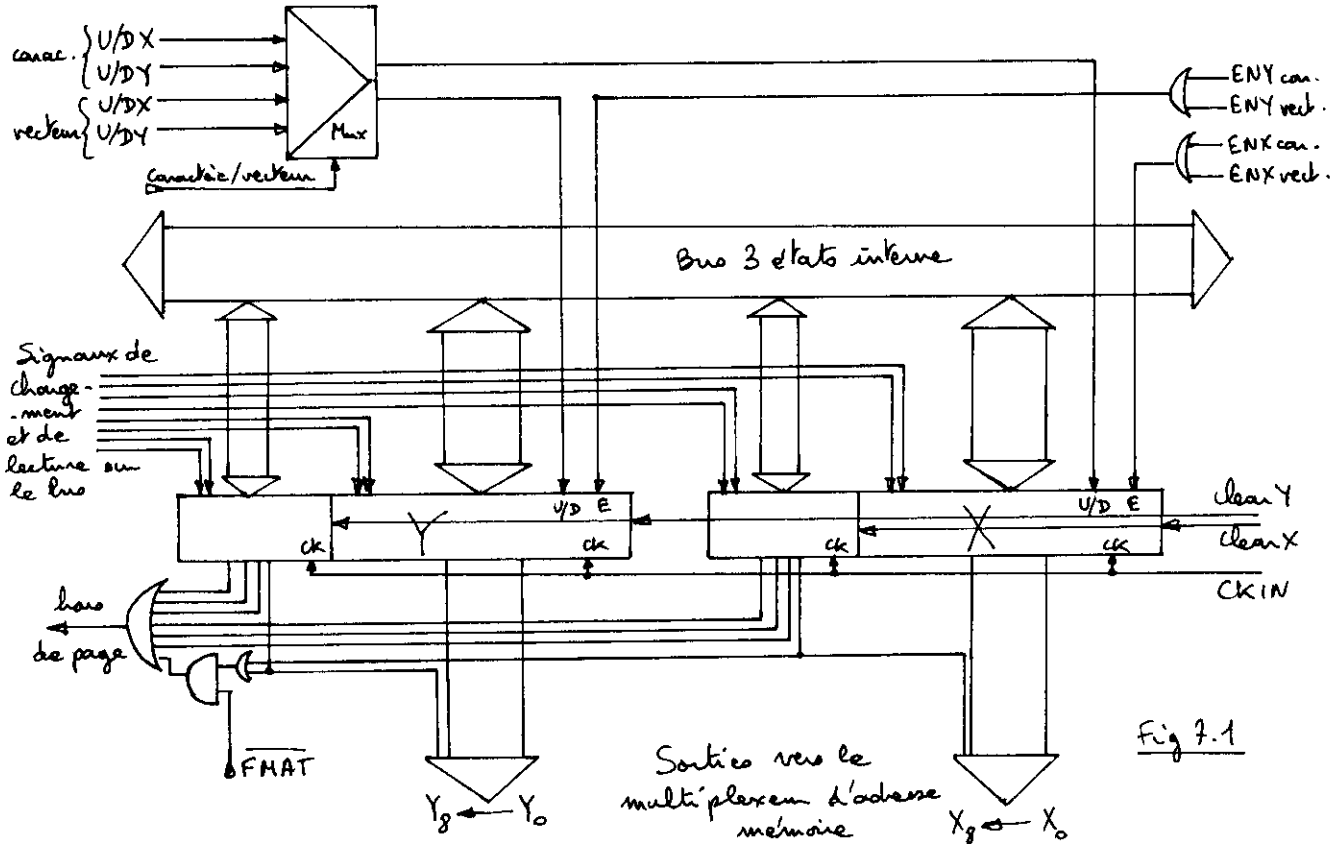


Fig 7.1

Les entrées Enable venant des générateurs de vecteurs et de caractères étant exclusives (à moins qu'un utilisateur lance simultanément le tracé d'un vecteur et d'un caractère -à ses risques et périls vu l'unicité du spot-), un simple OU suffit pour produire les Enables de commande des compteurs X et Y. Par contre, les commandes UP/DOWN ne comportant pas d'état majoritaire doivent être

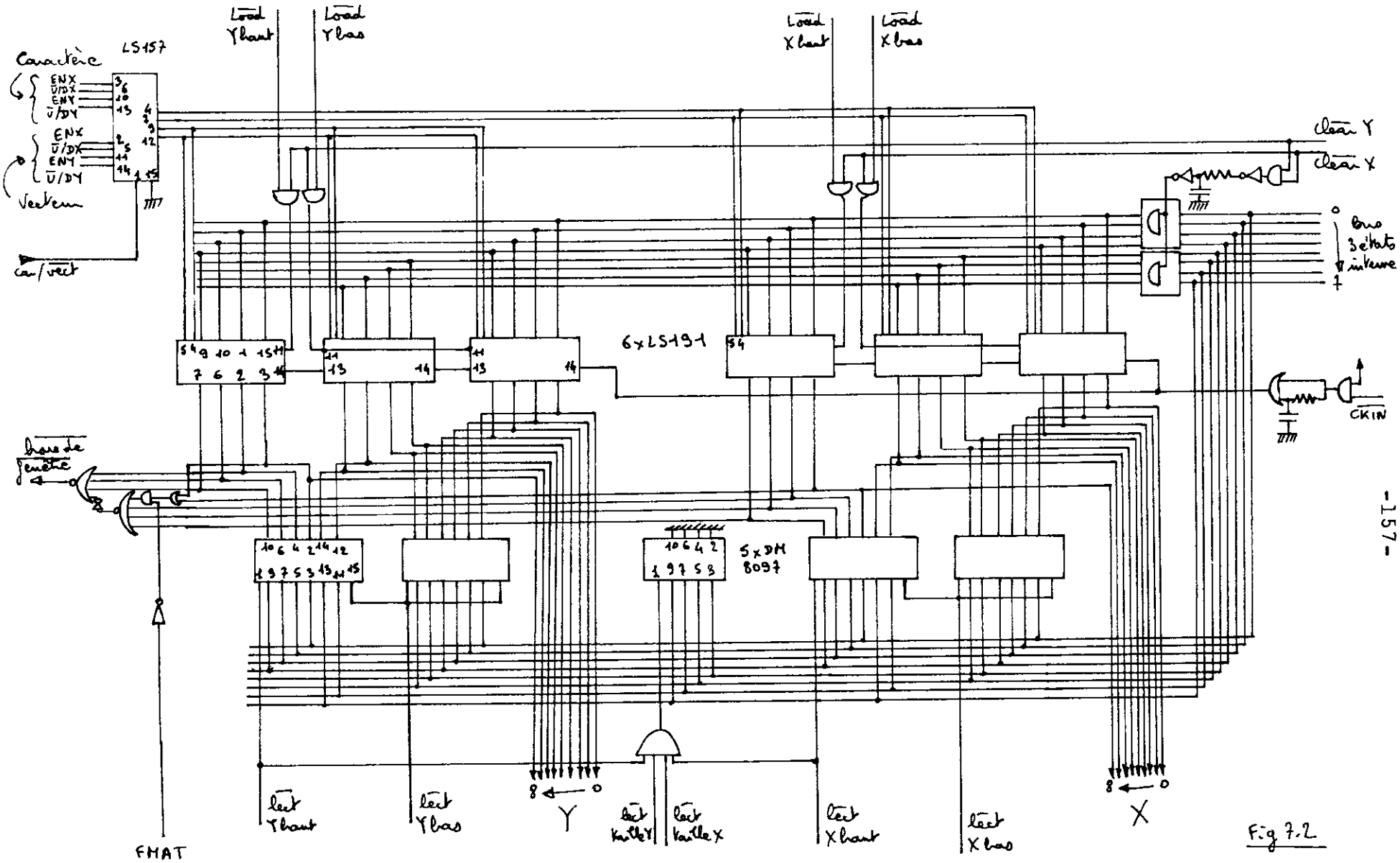

Caractère
LS157
ENX
U/DX
ENY
U/DY
Vect/vect
Load Y haut
Load Y bas
Load X haut
Load X bas
clear Y
clear X
Bus 3 état 8 entrée
6 x LS191
CKIN
horloge fenêtre
5 x DM 8097
FMAT
lect Y haut
lect Y bas
lect haut/bas Y
lect haut/bas X
lect X haut
lect X bas
8
Y
X

Fig 7.2

choisies par un multiplexeur commandé par un signal généré par le décodage des commandes.

Un reconnaisseur sur les adresses de poids fort (fonction de FMAT) indique si le spot est hors de la fenêtre affichée. Ce signal contrôle un bit du mot d'état ainsi que le signal $\overline{IWEN}$. La simulation MSI de l'ensemble de la figure 7.1 est indiquée en figure 7.2.

Du point de vue implantation, X et Y sont des compteurs UP/DOWN, avec Enable et commande UP/$\overline{DOWN}$, avec Clear synchrone, Load asynchrone, et positionnables sur le bus par une sortie 3 états. Une tranche de 2 bits de ces compteurs peut se réaliser à partir des briques 5, 8.5, 11.3, 11.4, et 2 (Fig.7.3).

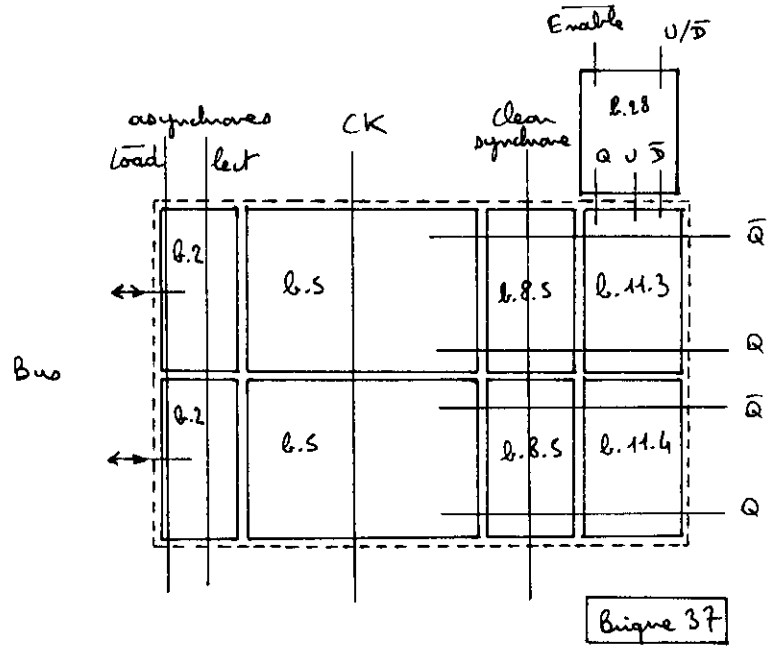


Fig 7.3

Les entrées Q,U,$\overline{D}$ sont commandées par une brique 28 symétrisée. L'ensemble des 2 compteurs et des circuits annexes est représenté figure 7.4.

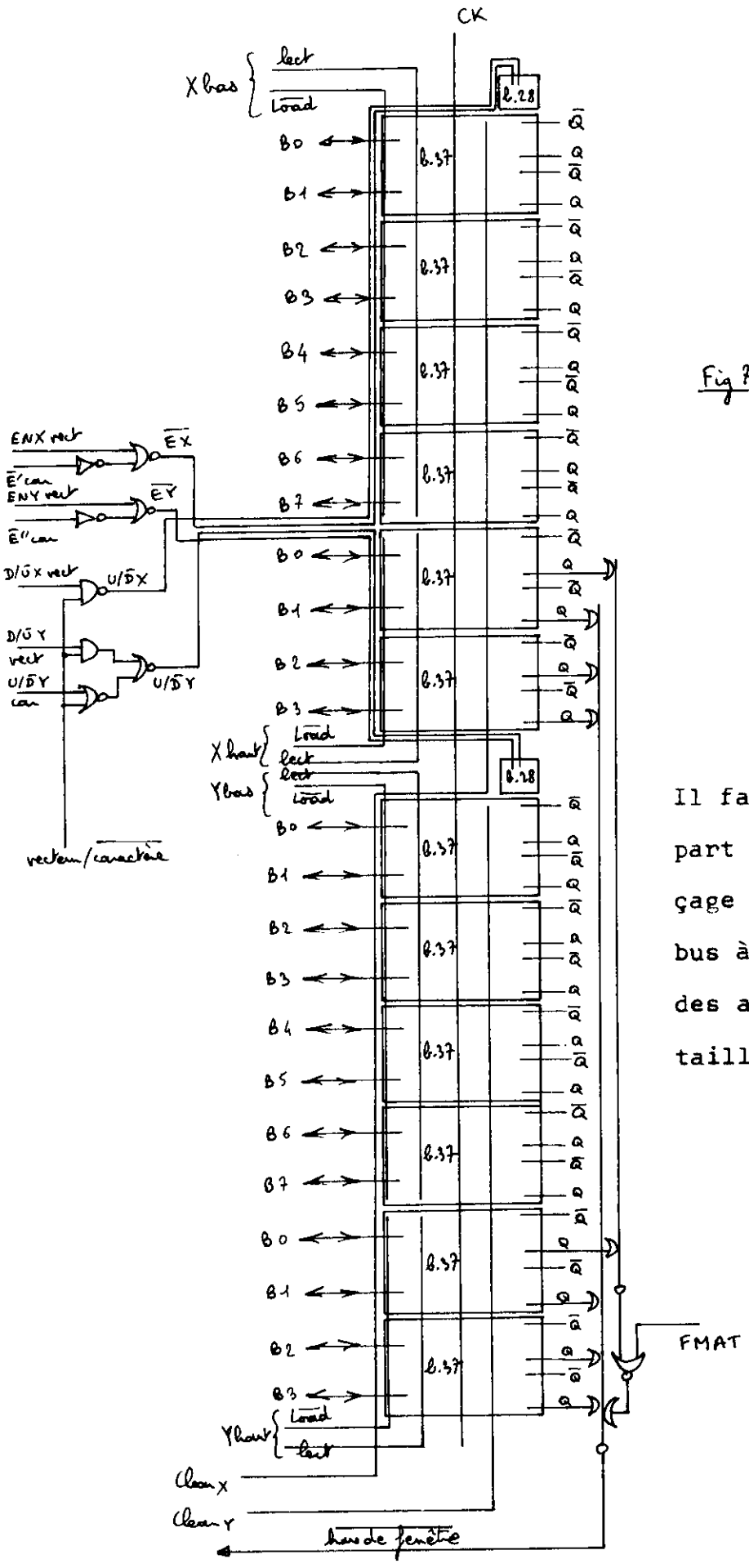
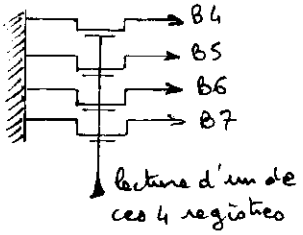


Fig 7.4

Il faut adjoindre quelque part 4 transistors de forçage des 4 bits forts du bus à 0 en cas de lecture des adresses Xhaut, Yhaut, tailleXcar, tailleYcar.



7.2-REGISTRE DE CONTROLE, MOT D'ETAT ET SYSTEME D'INTERRUPTION

a)Registre de contrôle. C'est un latch 8 bits, accessible en lecture et en écriture sur le bus bidirectionnel. Il commande le type de trait des vecteurs, contrôle les sorties IDIN et $\overline{\text{IWEN}}$ (mode Marquant/effaçant et spot Allumé/éteint), contient le masque des interruptions, et contrôle la dimension de l'espace adressable (Tableau 2.1).

Le contrôle de la sortie IDIN est très simple: cette sortie est identique à l'inverse du bit 3 du registre de contrôle (Marquant/$\overline{\text{effaçant}}$), à moins qu'elle ne soit forcée à "1" pour l'effacement de l'écran:

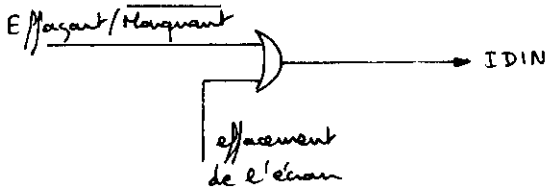


Fig 7.5

$\overline{\text{IWEN}}$ est bas lorsque l'une des sorties "spot" des générateurs de vecteurs ou de caractères est haut, à condition que le spot soit allumé (bit 2 du mot de contrôle à "1"), et que le spot soit dans la fenêtre visible. Il peut être aussi forcé bas par l'effacement image ou le positionnement d'un fond (Fig.7.6):

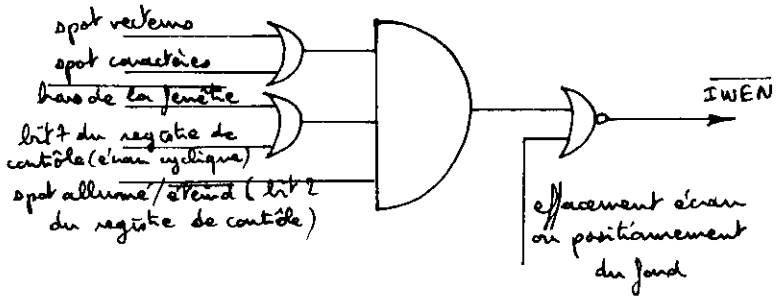


Fig 7.6

b)Signal "libre/occupé". Le système d'écriture peut être occupé pour 4 causes différentes: l'activité du générateur de vecteurs, celle du générateur de caractères, celle du mécanisme d'effacement

de l'image (ou de positionnement d'un fond) et enfin par l'attente de prise d'effet d'une commande (temps de décodage et de synchronisation avec l'horloge CKIN).

L'exécution d'une commande "longue" (c'est à dire qui entraînera une des 3 premières causes), se passe de la façon suivante: tout d'abord, l'écriture à l'adresse 0 arme le mécanisme de synchronisation des commandes qui attend pour l'exécution la première période d'écriture complète (voir §7.5). Pendant ce temps, le système d'écriture est signalé comme occupé (c'est la 4ème cause citée ci-dessus). Puis, quand la synchronisation est obtenue, la cause d'interruption change.

Dans le cas d'exécution de commandes courtes, (tel que "Clear registre X" par exemple), seule l'attente de la synchronisation "occupe" le système d'écriture.

Les signaux "occupé" des 3 premières causes sont toujours cadrées entre 2 fronts descendant de CKIN, mais ne sont pas dépourvues d'éventuels parasites. La 4ème cause est "propre", mais son début ne coïncide pas avec un front descendant de CKIN. Le signal "libre/occupé"est généré par le montage de la figure 7.7:

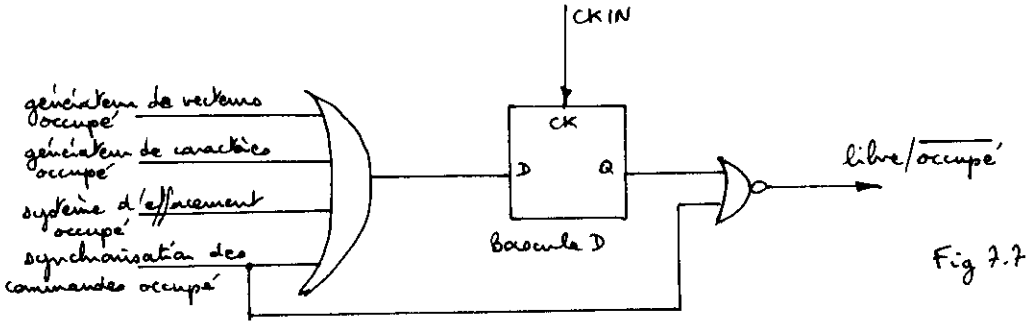


Fig 7.7

c)Mécanisme de contrôle des interruptions et mot d'état. 3 signaux internes peuvent provoquer une interruption: système d'écriture libre, fin de trame, light-pen terminé. Ces signaux sont accessibles dans le mot d'état (bits 0,1,2) qui est constitué de 8 barrières

3 états.

Ces causes d'interruption mettent chacune à "1" une bascule sur leur front montant. La lecture du mot d'état remet à zéro ces bascules. Ceci permet de supprimer une interruption sans avoir supprimé sa cause et sans l'avoir masquée, par exemple si on désire attendre la suivante de même nature.

Les sorties de ces 3 bascules sont accessibles dans le mot d'état (bits 4,5,6) après être passées à travers le masque d'interruption. Le OU logique de ces 3 bits est lui aussi accessible (bit 7). C'est son inverse qui sort sur $\overline{\text{MIRQ}}$ (sortie à collecteur ouvert) voir figure 2.27.

La remise à zéro des bascules ne doit pas se faire systématiquement lors de la lecture du mot d'état, car alors on risquerait de "perdre" une interruption (Fig.7.8). Il faut échantillonner la bascule dans un premier temps, puis la remettre à zéro dans un deuxième temps, conditionnellement au résultat de l'échantillonnage (Fig.7.9).

Fig 7.8

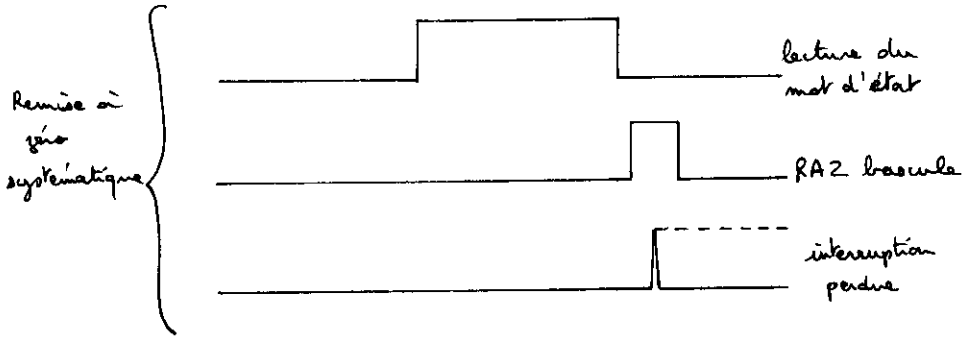
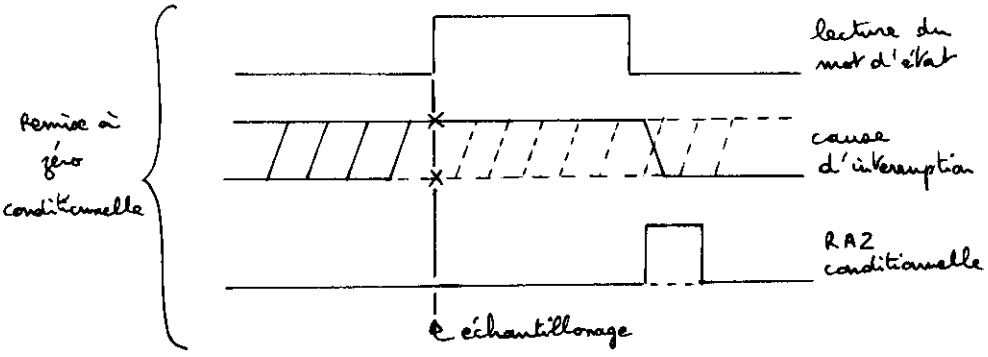


Fig 7.9



Cette remise à zéro conditionnelle peut se réaliser par le montage de la figure 7.10:

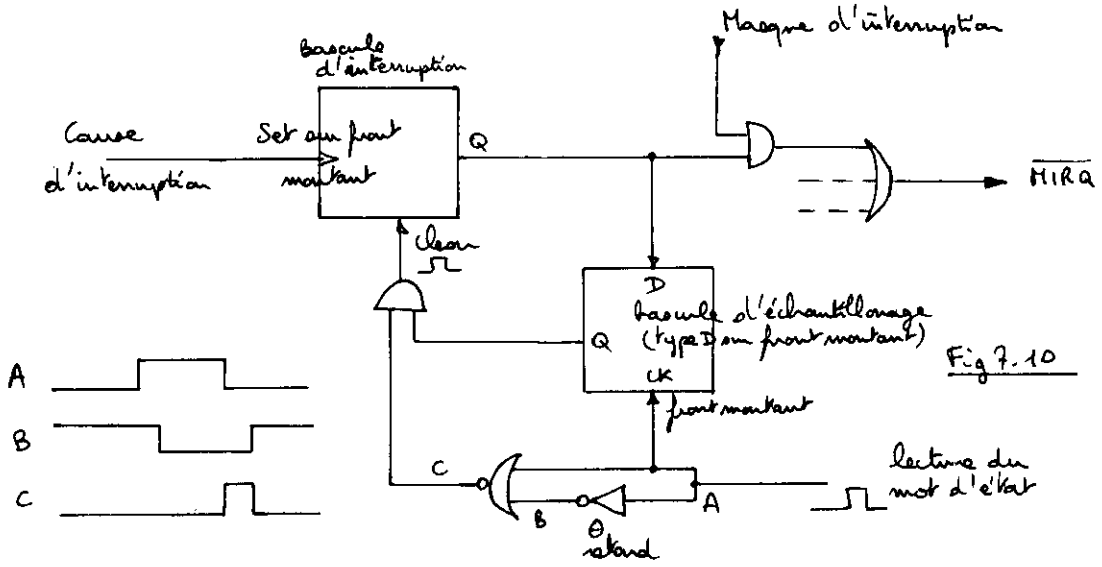


Fig 7.10

Le bit restant du mot d'état (bit 3) indique si le spot est hors de la fenêtre affichée.

d)<u>Simulation MSI et implantation.</u> L'ensemble décrit dans ce paragraphe 7.2 peut être simulé par le montage de la figure 7.11.

Du point de vue implantation, le registre de contrôle est un latch réalisé à partir de briques 3.1. Un de ses bits, associé à un bit du mot d'état est représenté figure 7.12.

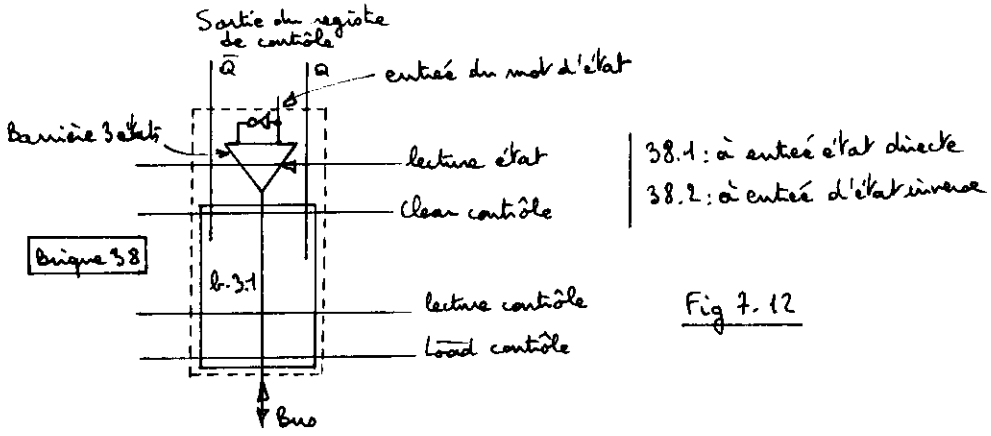


Fig 7.12

L'ensemble bascule d'interruption-bascule d'échantillonnage et masque pour une cause d'interruption est représenté figure 7.13.:

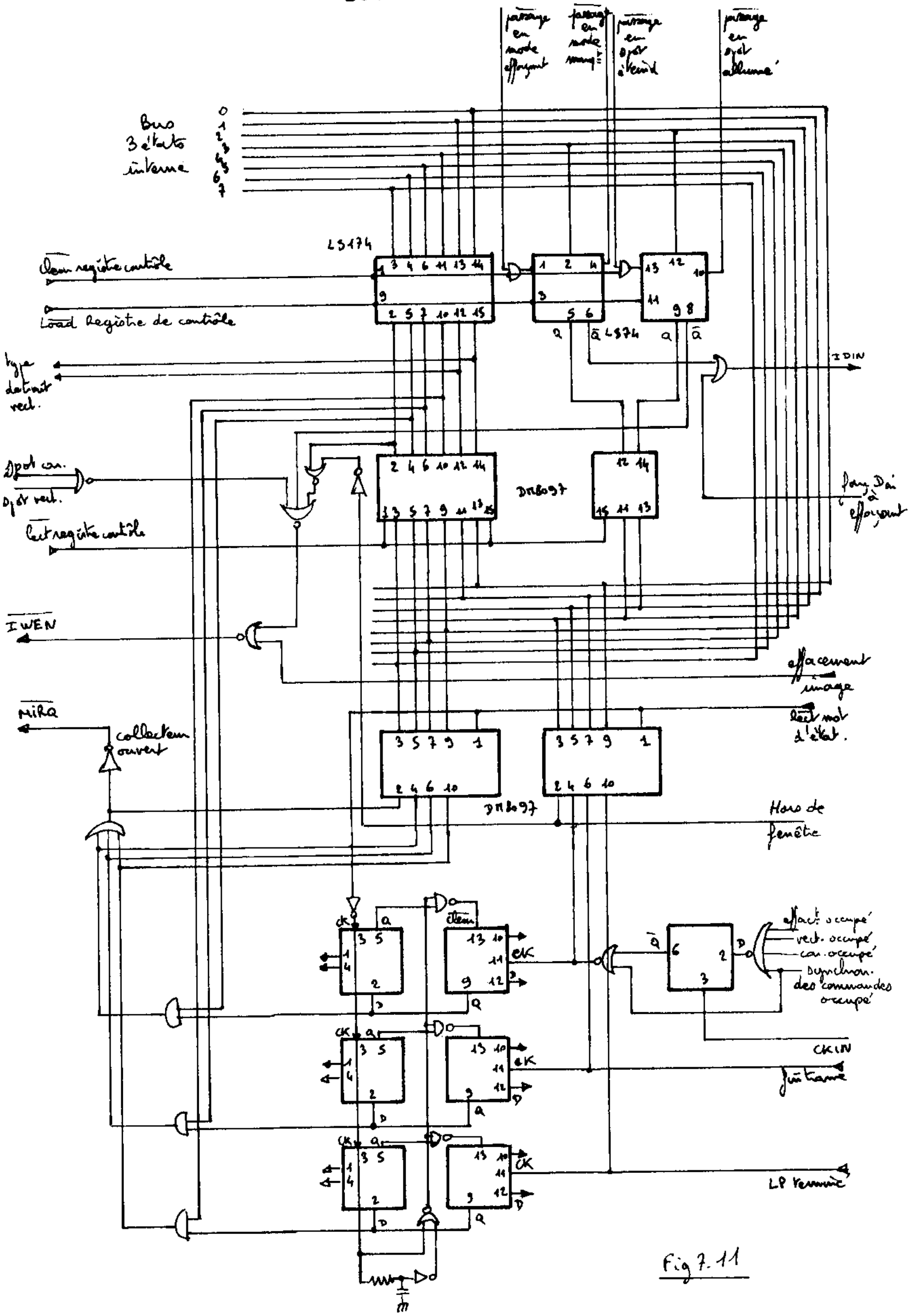

passage en mode effaçant
passage en mode marqué
passage en spot éteint
passage en spot allumé
Bus 3 états interne
0 1 2 3 4 5 6 7
LS174
Clear registre contrôle
Load Registre de contrôle
type de trait vect.
Spot cur.
Spot vect.
Lect registre contrôle
DM8097
IWEN
MIRQ
collecteur ouvert
IDIN
effacement image
lect mot d'état
Hors de fenêtre
effact occupé
vect. occupé
cur. occupé
synchron. des commandes occupé
CKIN
fin trame
LP revenue

Fig 7.11

L'ensemble décrit dans ce chapitre 7.2 peut s'implanter comme en figure 7.14.

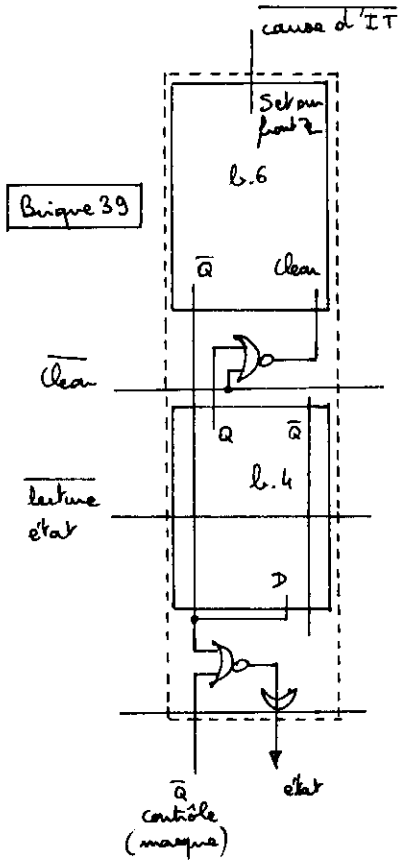


Fig 7.13

7.3-LOGIQUE DU LIGHT-PEN

Le fonctionnemnt interne est décrit par les signaux de la figure 2.25. La réalisation est donnée figure 7.15. La bascule "light pen non lu", remise à zéro par la lecture des registres Xlightpen ou Ylightpen doit être accompagnée d'une bascule d' échantillonnage, pour la même raison que les bascules d'interruption. La maquette de simulation est donnée figure 7.16 et l'implantation figure 7.17. Les registres light-pen sont constitués de latchs à sorties 3 états (Fig.7.17.a).

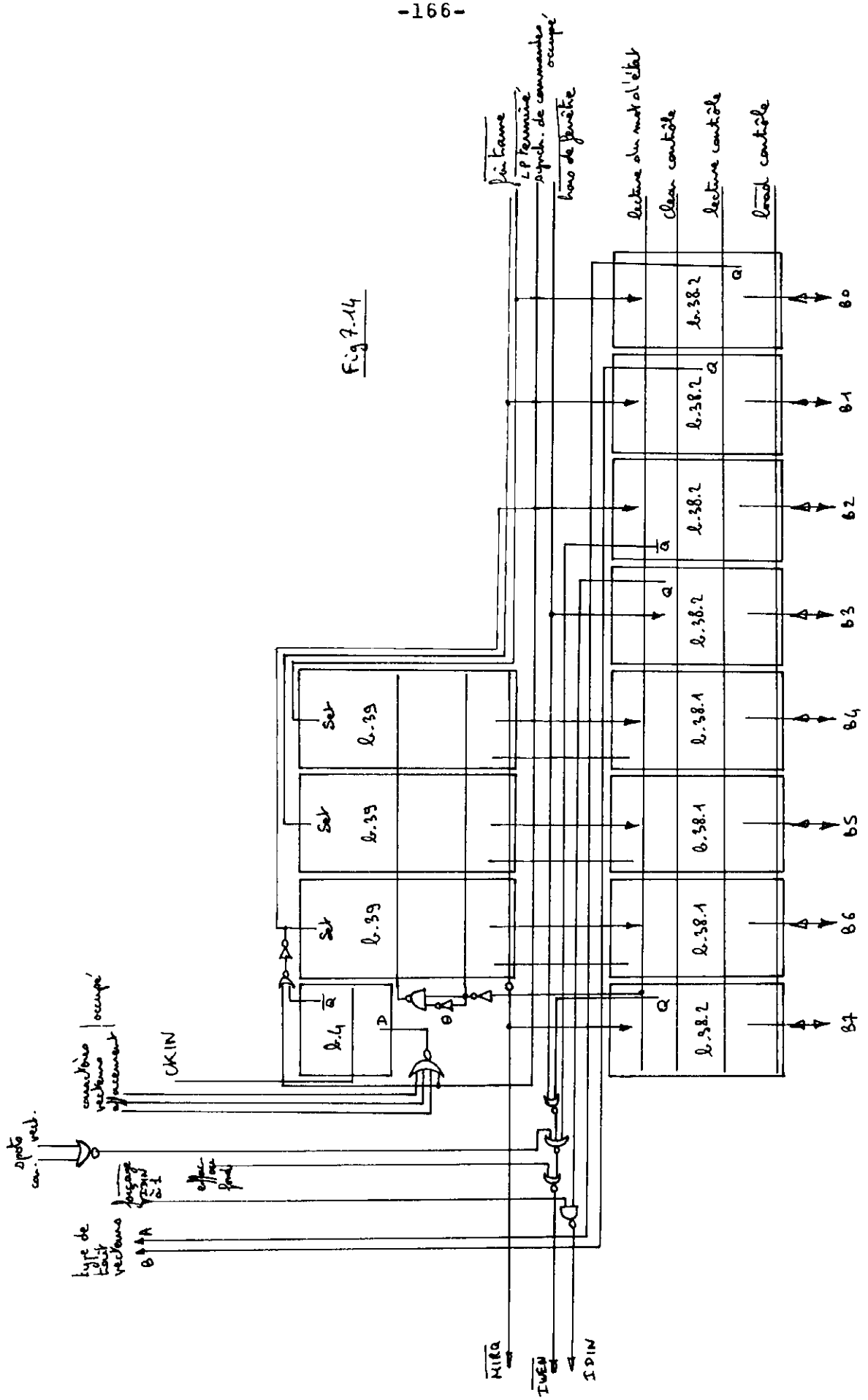


Fig 2.14

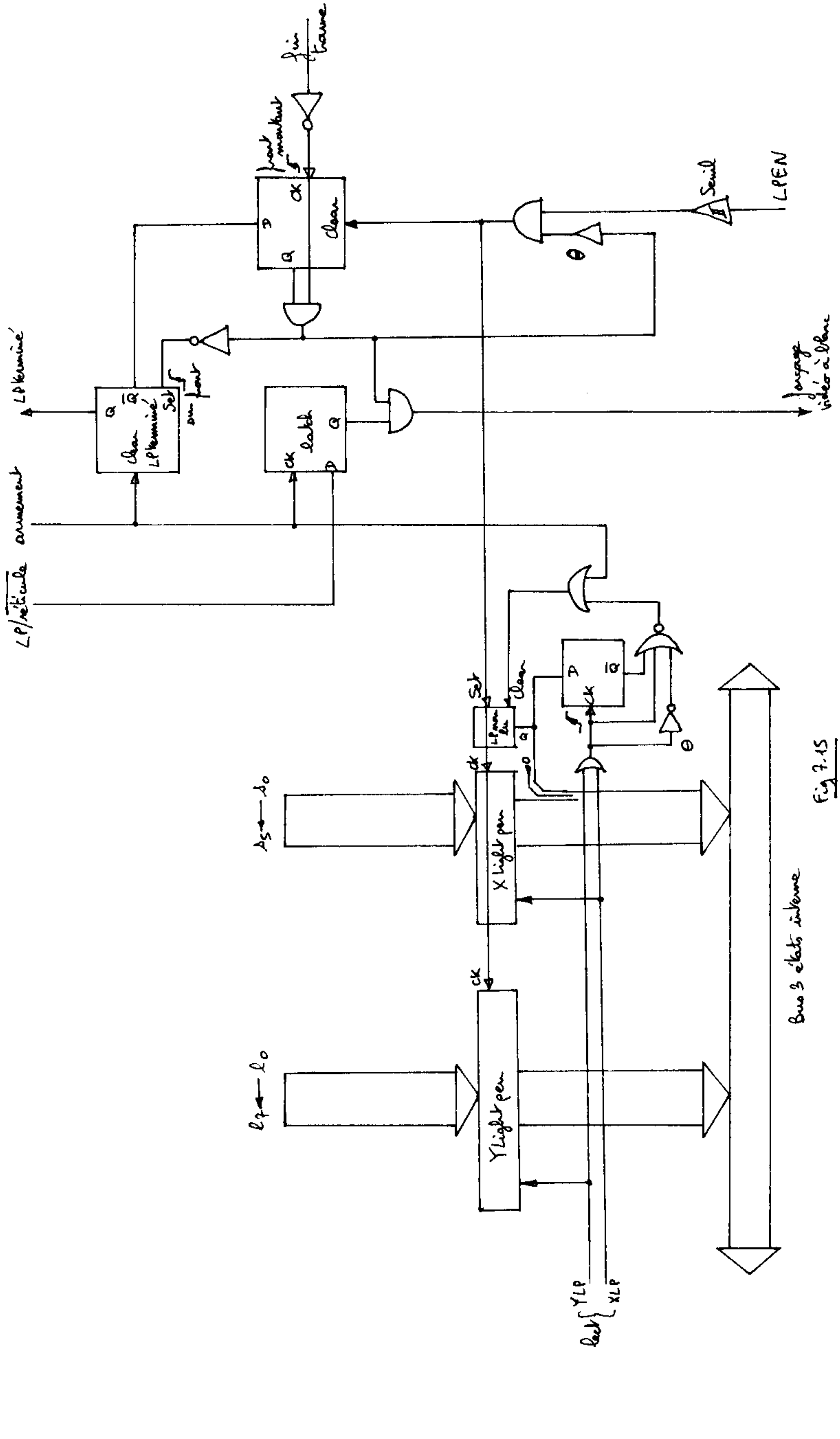

fin trame
Seuil
LPEN
clear
CK
latch
Set
clear
LP terminé
armement
LP/réticule
X light pen
Y light pen
Bus 3 états interne
YLP
XLP

Fig 7.15

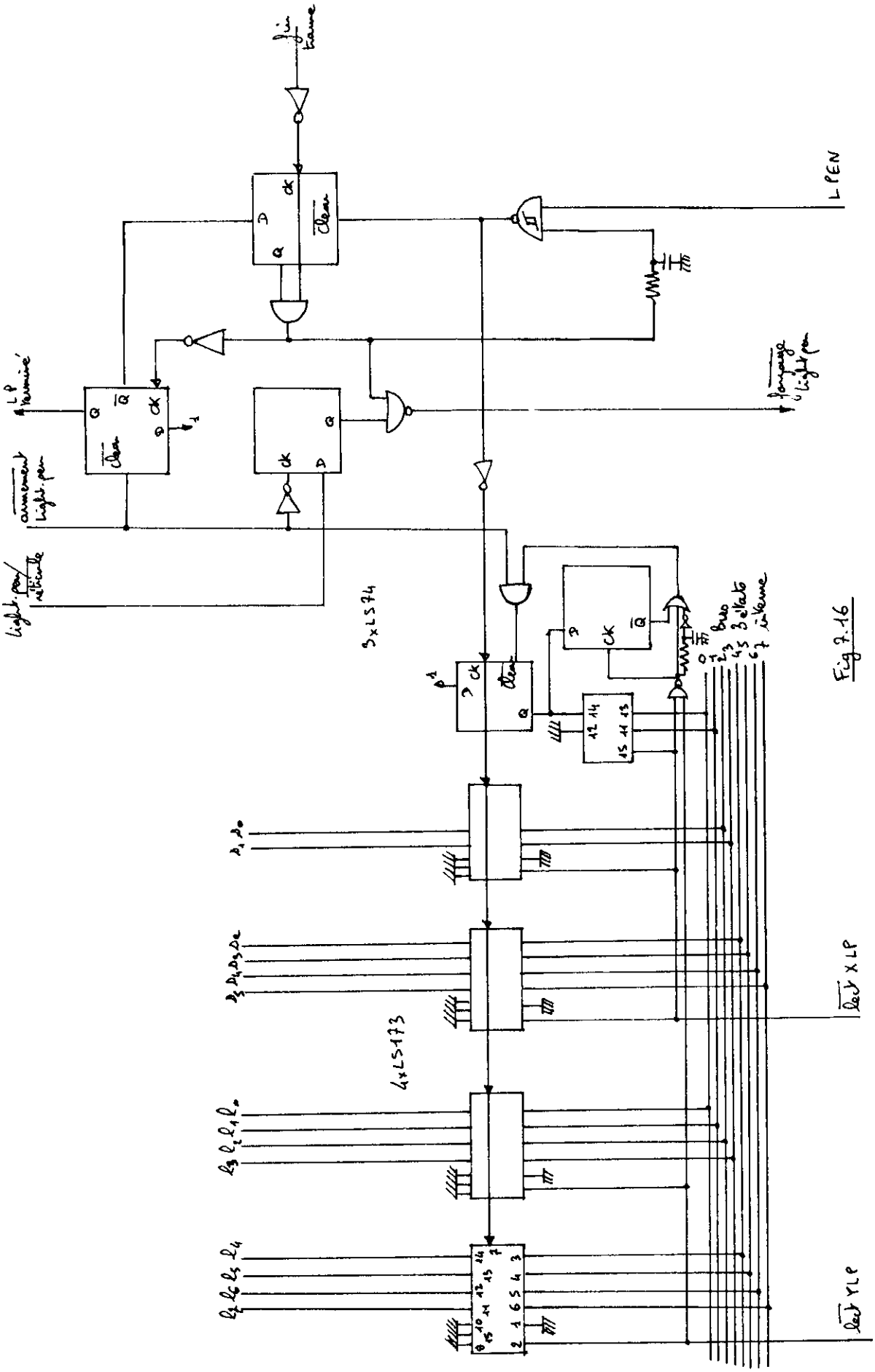


Fig 2.16

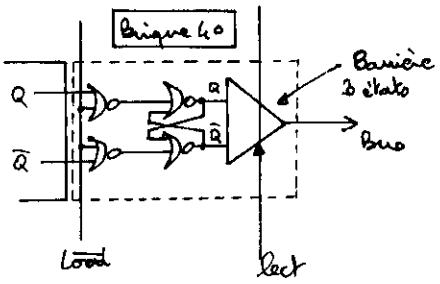


Fig 7.17.a

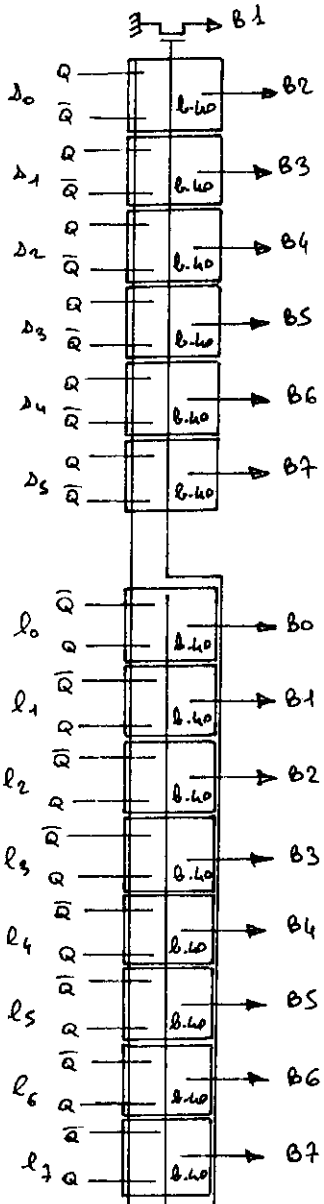
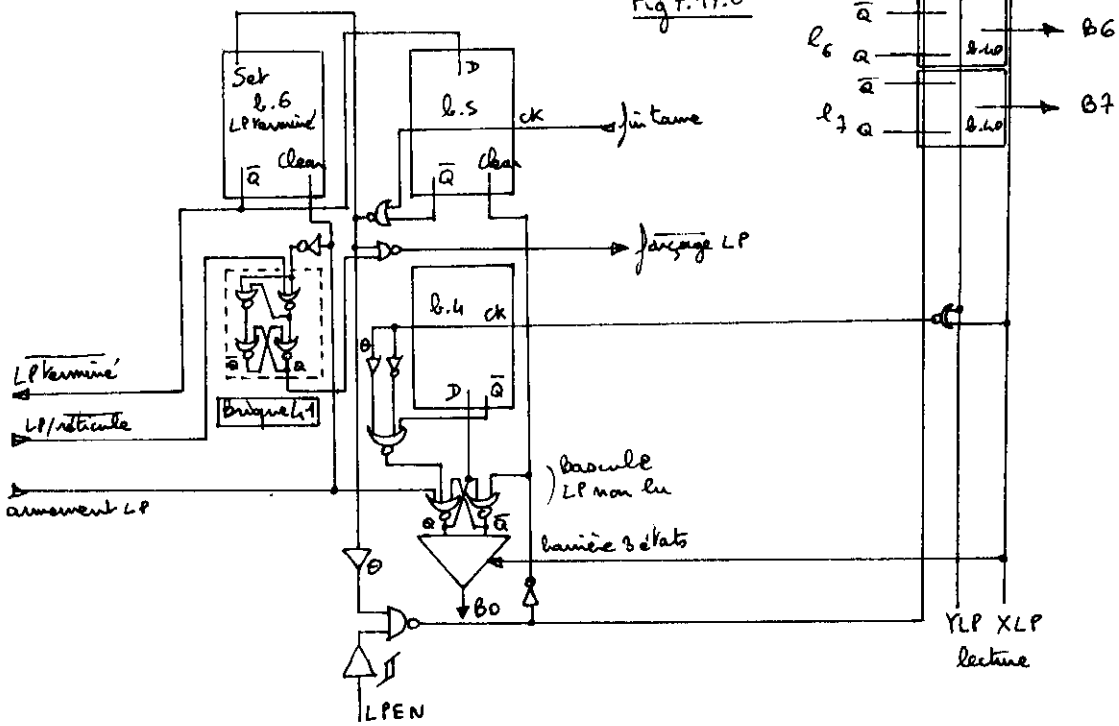


Fig 7.17.b



7.4-EFFACEMENT DE L'IMAGE OU POSITIONNEMENT D'UN FOND

Le fonctionnement de ce mécanisme est décrit paragraphe 2.14. Sa réalisation pratique est la suivante: un compteur 2 bits basculant sur le front descendant de "fin trame" est généralement bloqué sur la valeur 3. La commande d'effacement charge (de façon asynchrone) ce compteur à $01_{(2)}$ si FMAT est bas, à $00_{(2)}$ si FMAT est haut. Le système d'effacement est actif tant que l'état n'est pas $11_{(2)}$:

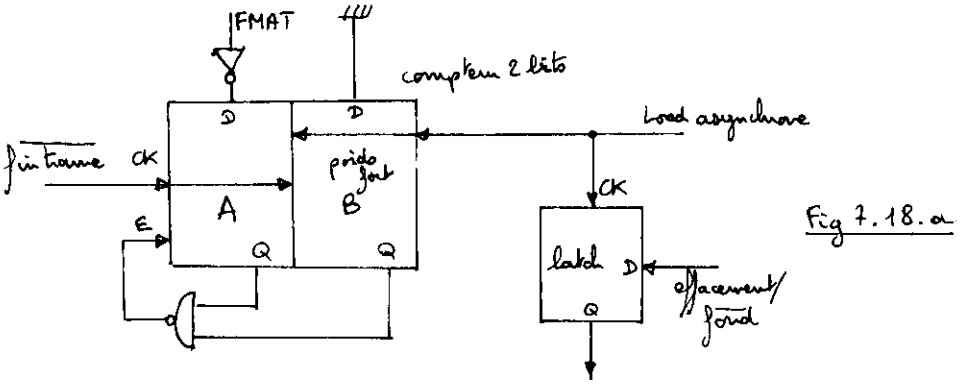


Fig 7.18.a

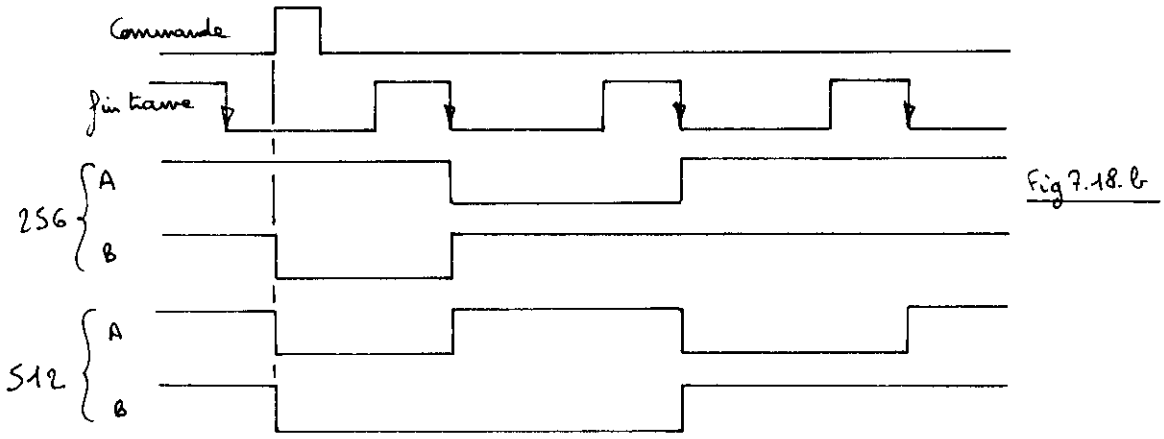


Fig 7.18.b

"Occupé" est le Enable de ce compteur sauf si fin trame est haut alors que A est bas et B est haut. "Forçage $\overline{\text{IWEN}}$ à 0" est ce Enable sauf si fin trame est haut, "Forçage IDIN à 1" aussi, mais avec la condition effacement/$\overline{\text{fond}}$.

La simulation MSI de ce mécanisme est donnée figure 7.19 et son implantation figure 7.20.

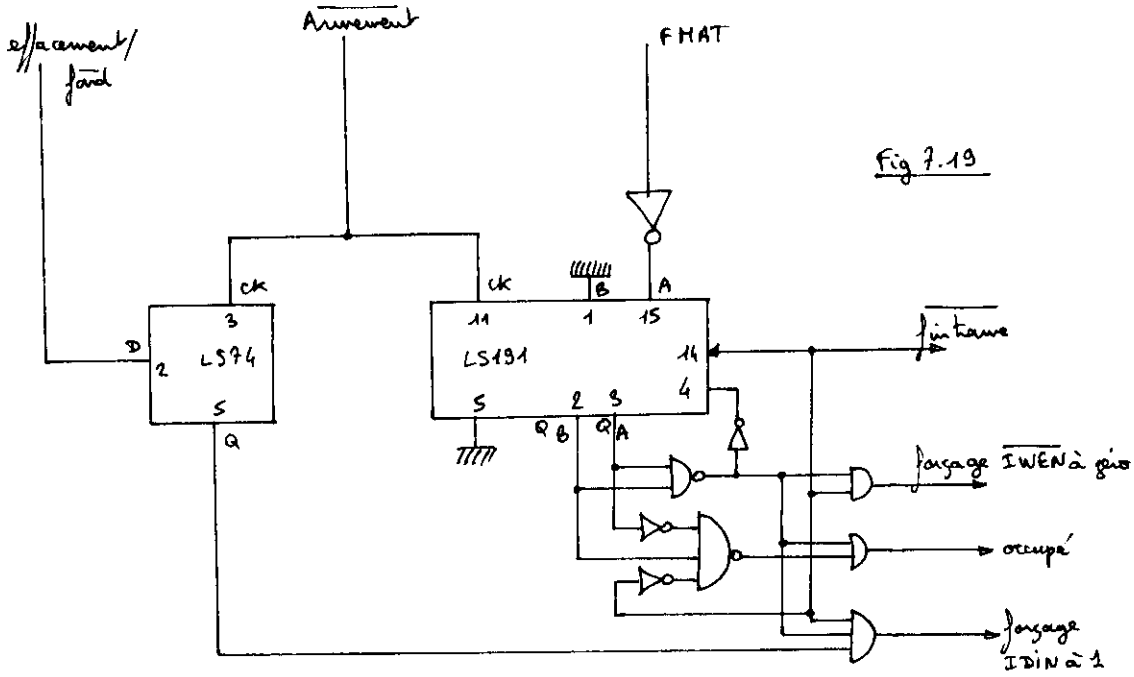


Fig 7.19

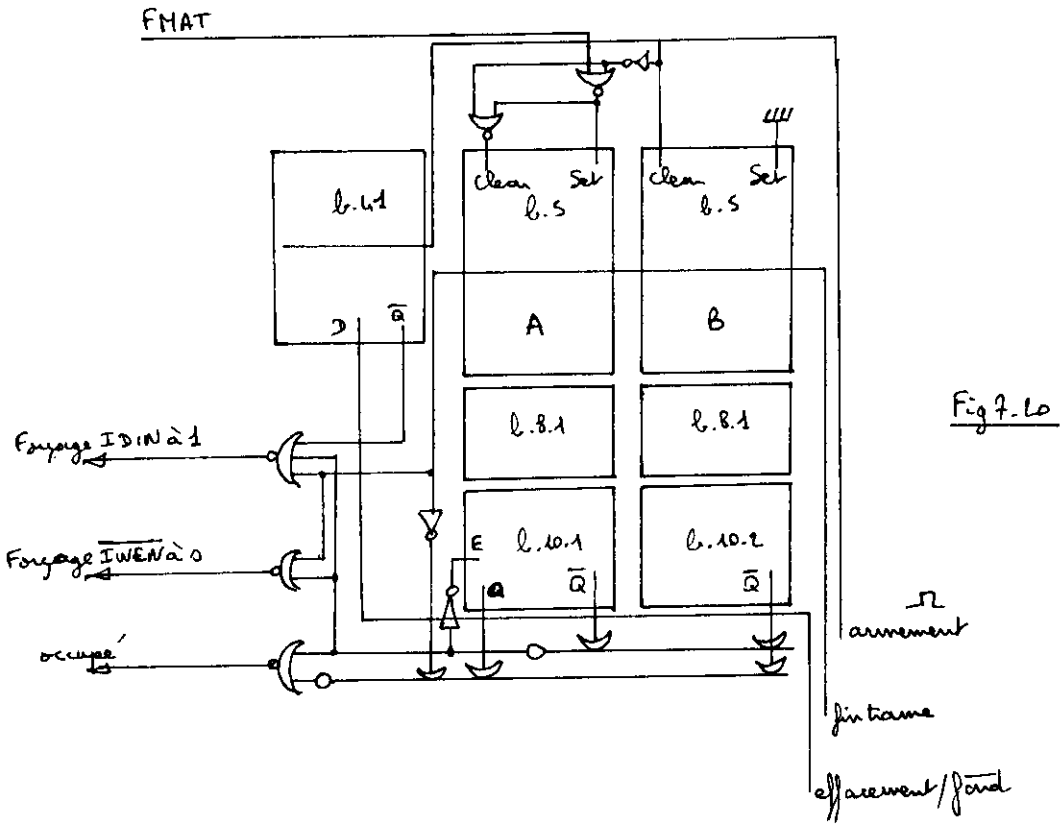


Fig 7.20

7.5-REGISTRE DE COMMANDES, DECODAGE ET SYNCHRONISATION

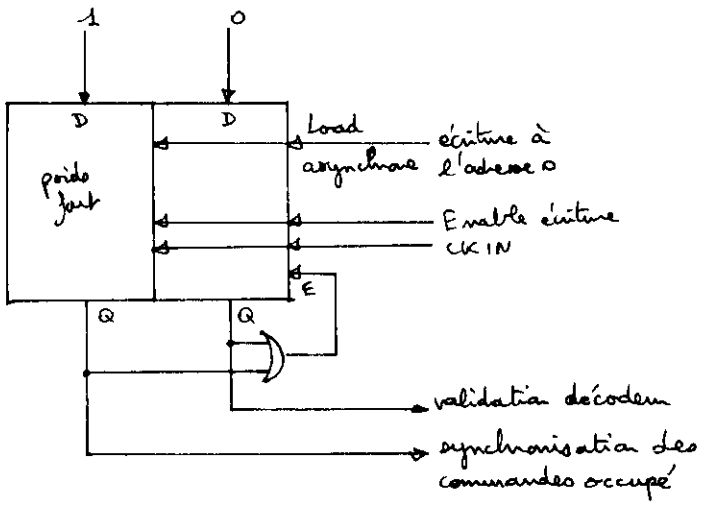
Le registre de commandes est un latch de 8 bits accessible uniquement en écriture, à l'adresse 0. Ses sorties attaquent un décodeur constitué entre autres de la ROM générateur de caractères.

Les commandes peuvent se diviser en 2 groupes:"statiques" et "dynamiques".

-Les commandes statiques sont de simples fonctions combinatoires du registre de contrôle et agissent comme paramètres sur les différents automates, par exemple: direction d'un vecteur, "effacement/ $\overline{\text{fond}}$", etc...

-Les commandes dynamiques sont les "armements" des différents générateurs. Elles sont activées pendant une période d'horloge où tout le système d'écriture est validé. Un système de synchronisation constitué d'un compteur 2 bits choisie la première période d'horloge qui suit l'accès à l'adresse 0 (chargement du registre de commandes). Son montage est donné figure 7.21. La sortie du bit faible est le signal de validation du décodeur de commandes dynamiques. L'état de blocage est l'état 0. Le lancement se fait par un Load asynchrone prioritaire.

Fig 7.21.a



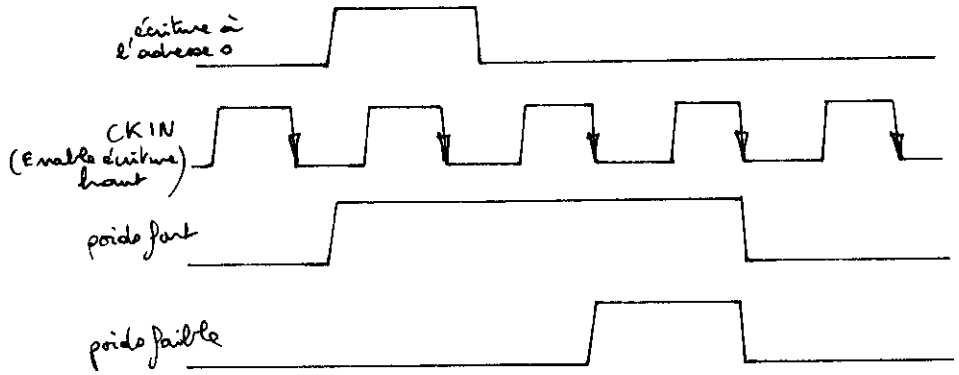


Fig 7.21.b

La classification des commandes en 2 groupes est la suivante:

commandes statiques:

-direction des vecteurs,

-petits vecteurs,

-vecteurs de directions privilégiées,

-$|\Delta X|$, $|\Delta Y|$ petit vecteur,

-code caractère,

-pavés,

-pavé 4x4,

-light-pen/$\overline{\text{réticule}}$,

-effacement/$\overline{\text{fond}}$,

-vecteur/caractère.

commandes dynamiques:

-lancement vecteur,

-lancement caractère,

-armement light-pen ou réticule,

-effacement ou fond,

-Clear registre X,

-Clear registre Y,

-mode effaçant,

-mode marquant,

-spot allumé,

-spot éteint,

-réinitialisation,

-sortie $\overline{\text{LSTR}}$.

La réinitialisation comprend:

-Clear registre de contrôle,

-Clear X,

-Clear Y,

-Clear $|\Delta X|$, $|\Delta Y|$,

-Preset à 1 tailleXcar et tailleYcar,

-effacement image.

La simulation MSI de cette partie est donnée figure 7.22, et sa réalisation figure 7.23. Le registre de commandes est un latch uniquement chargeable par le bus constitué de briques 41. Les sorties de ce registre attaquent la ROM générateur de caractères. On peut placer les décodages des commandes à côté des décodeurs d'entrées de cette ROM.

7.6-BUS 3 ETATS INTERNE ET ECHANGES AVEC LE MICROPROCESSEUR

Les registres internes peuvent être chargés ou lus par le bus MPD, de façon totalement asynchrone avec le signal d'horloge CKIN. $\overline{\text{MPCE}}$ est le signal temporel d'échange. Le bus interne comprend 14 fils:

-données :8 fils

-adresses :4 fils

-Read/Write:1 fil

-$\overline{\text{MPCE}}$:1 fil.

Au niveau de chaque registre s'effectue le décodage nécessaire (Fig.7.24).

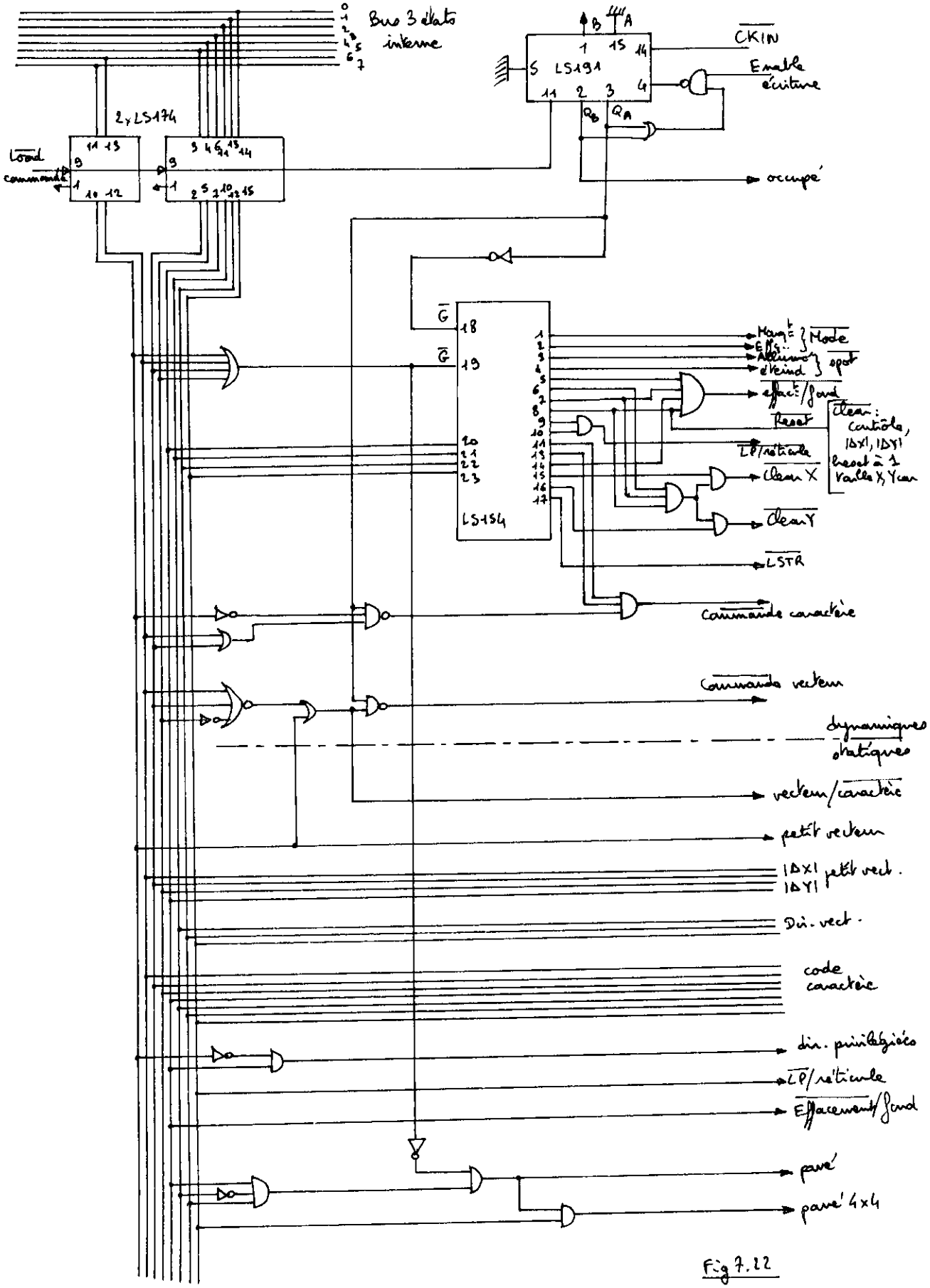
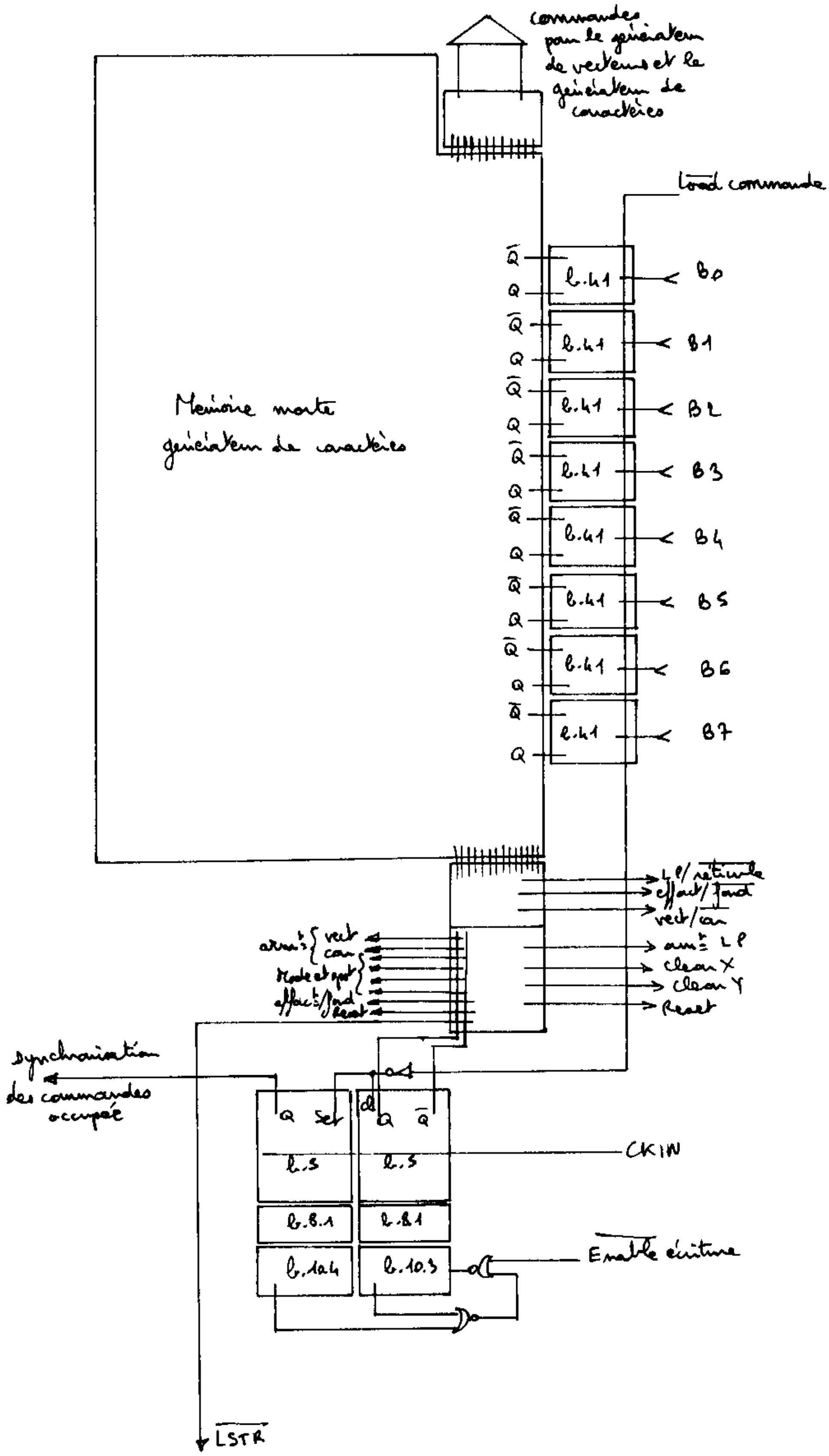

Bus 3 états interne
2 x LS174
Load commande
LS191
CKIN
Enable écriture
occupé
LS154
Mode
effacement/fond
Reset
clear : contrôle, IΔXI, IΔYI
reset à 1 valide X, Y car
LP/réticule
Clear X
Clear Y
LSTR
Commande caractère
Commande vecteur
dynamiques
statiques
vecteur/caractère
petit vecteur
IΔXI petit vect.
IΔYI
Dir. vect.
code caractère
dir. prioritaires
LP/réticule
Effacement/fond
pavé
pavé 4x4

Fig 7.22

Fig 7.23


commandes pour le générateur de vecteurs et le générateur de caractères
Load commande
Mémoire morte générateur de caractères
B0
B1
B2
B3
B4
B5
B6
B7
Reset
Clear X
Clear Y
synchronisation des commandes occupée
CKIN
Enable écriture
LSTR

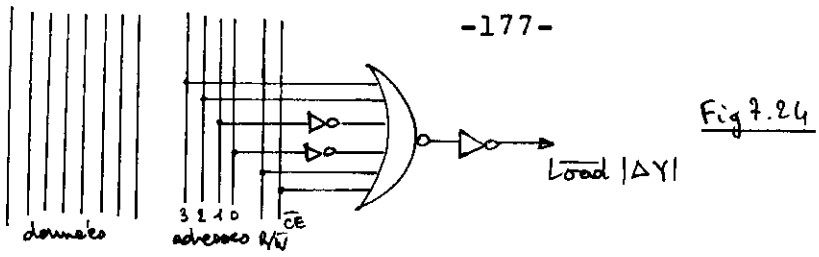


Fig 7.24

Le signal $\overline{ISTR}$ est généré par un tel décodeur (Fig.7.25).

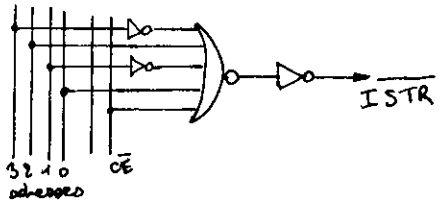


Fig 7.25

La simulation MSI de ce bus est donnée figure 7.26, et la répartition des sous-ensembles ainsi que le trajet du bus sur la puce Fig.7.27.

7.7-DISPOSITION DES SOUS-ENSEMBLES SUR LA PUCE

Il nous reste maintenant à rassembler toutes les parties implantées séparément, de telle manière que soient évitées les longues connexions, et à obtenir un ensemble rectangulaire, carré si possible.

1°)Tout d'abord, la succession des tranches:

-compteurs X,Y (pointeur d'écriture),

-multiplexeurs d'adresse mémoire,

-compteur de visualisation-synchronisation,

-registres light-pen,

nous est imposée (voir Fig.4.19), ainsi que le parcours en U du bus autour de ce bloc, vu les connexions avec les registres X,Y d'une part, light pen d'autre part. Les adresses mémoire d'image sortent alors au-dessus de ce U (Fig.7.27).

2°) Vu la longueur de ce bloc, il est obligatoire de disposer les parties écriture à droite ou à gauche (Fig.7.27). Nous avons choisi à gauche pour raccourcir les connexions de commande des registres X et Y. Ce même argument nous impose à peu près la disposi-

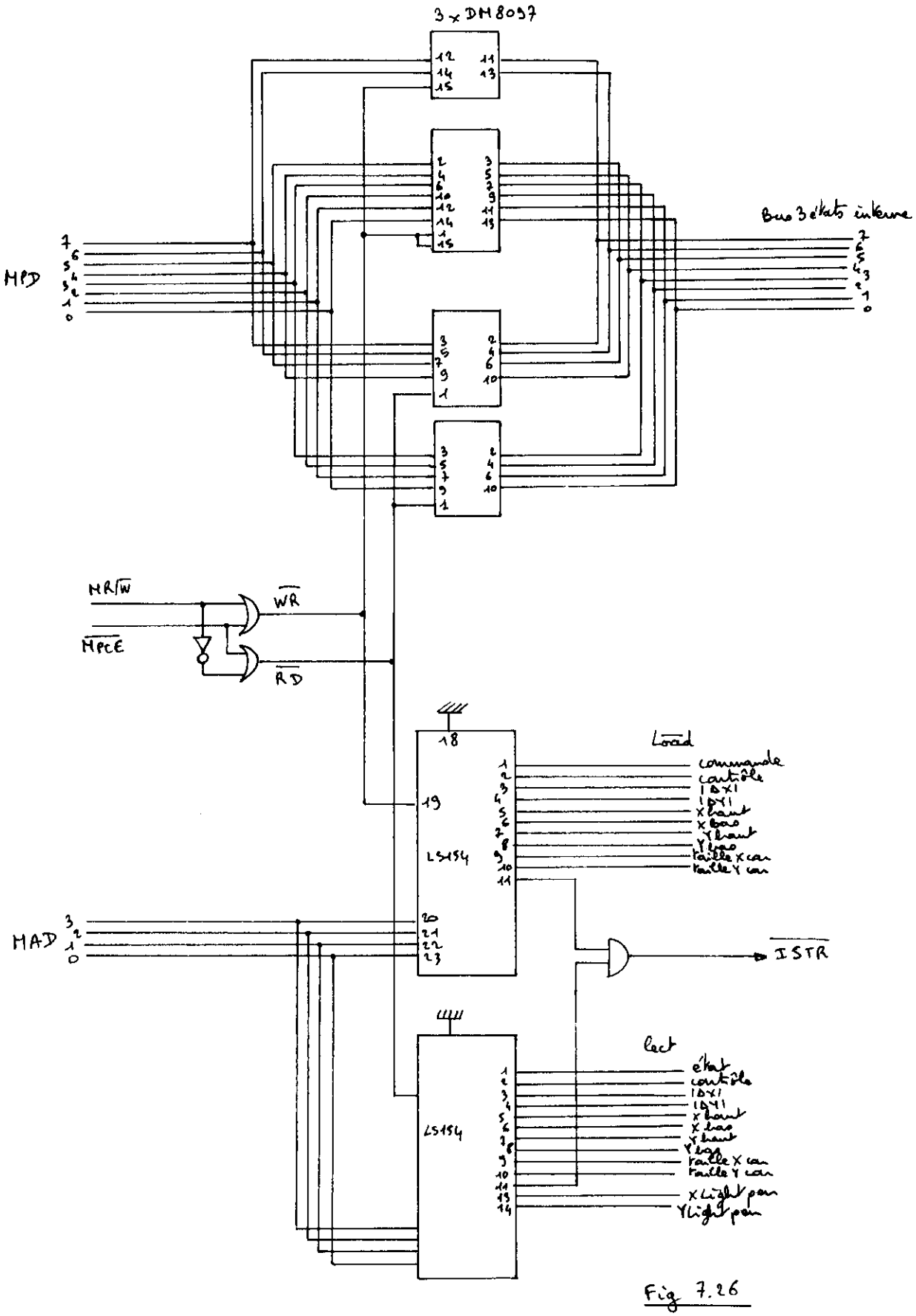

3 x DM8097
Bus 3 états interne
MPD
MR/W
WR
MPCE
RD
Load
commande
contrôle
IDX1
IDY1
X haut
X bas
Y haut
Y bas
taille X car
taille Y car
LS154
MAD
ISTR
lect
état
contrôle
IDX1
IDY1
X haut
X bas
Y haut
Y bas
taille X car
taille Y car
X Light pen
Y Light pen
LS154

Fig 7.26

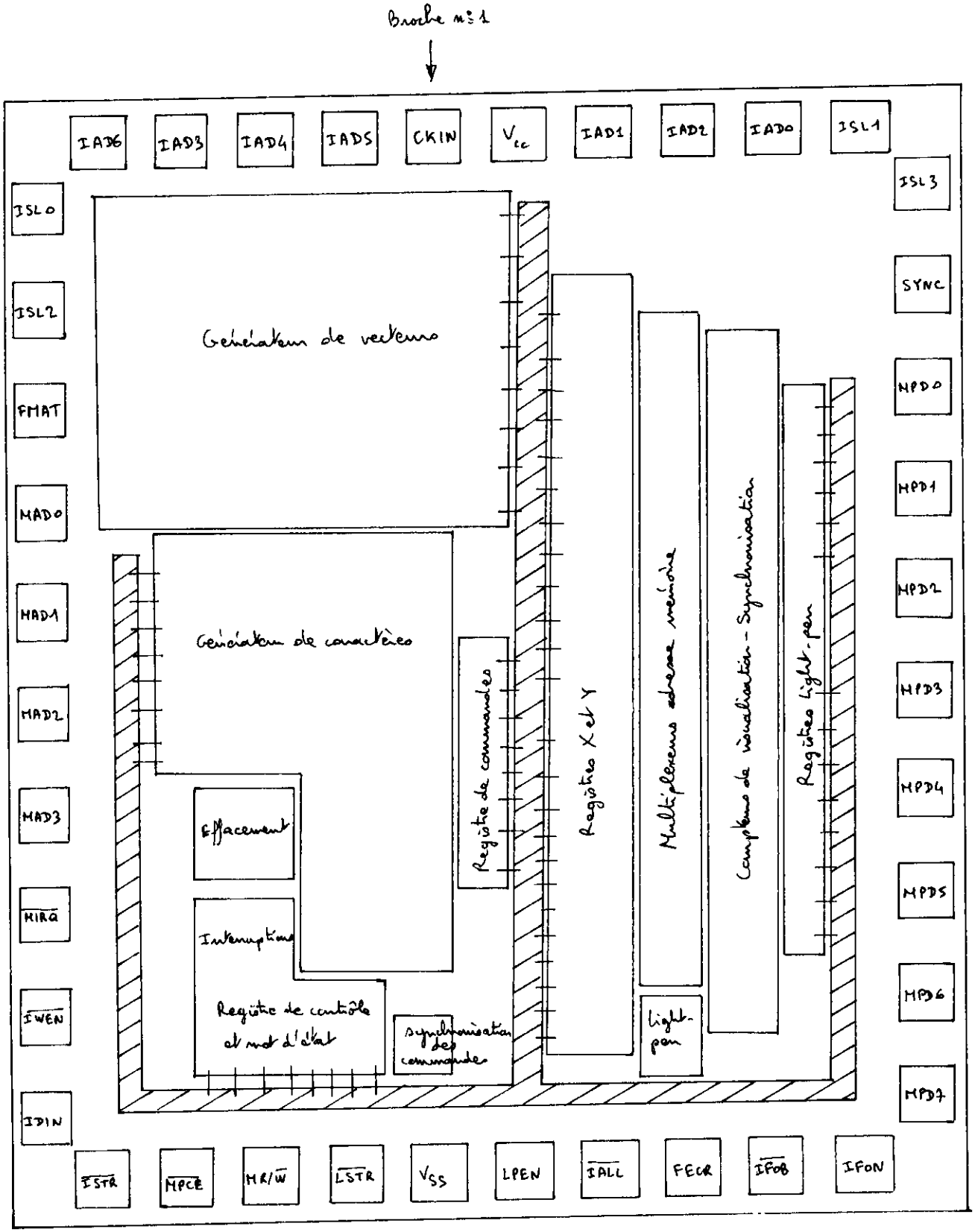

Broche n°1
IAD6
IAD3
IAD4
IAD5
CKIN
V_{cc}
IAD1
IAD2
IAD0
ISL1
ISL3
ISL0
SYNC
ISL2
Générateur de vecteurs
NPD0
FMAT
NPD1
MAD0
NPD2
MAD1
Générateur de caractères
NPD3
MAD2
Registre de commandes
Registres X et Y
Multiplexeur adresse mémoire
Compteur de visualisation - Synchronisation
Registres light-pen
NPD4
MAD3
Effacement
NPD5
MIRQ
Interruptions
Registre de contrôle et mot d'état
Synchronisation des commandes
light-pen
NPD6
IWEN
NPD7
IDIN
ISTR
MPCE
MR/W
LSTR
V_{SS}
LPEN
IALL
FECR
IFOB
IFON

Fig 7.27

tion relative des blocs générateur de caractères et de vecteurs puisque les sorties de l'un sont en haut et de l'autre en bas. Il reste à caser les autres petits automates, en cherchant à minimiser le parcours du bus. La disposition proposée Fig.7.27 permet de ne transmettre que 4 bits de données dans la branche verticale gauche du bus.

3°)Il reste alors à positionner les plots, si possible 10 par côté de ce rectangle. La disposition de la Fig.7.27 permet les caractéristiques suivantes:

-sorties IAD dans des positions analogues aux broches des mémoires 16K et 4K,

-broches MAD et MPD aux mêmes emplacements que dans la famille du microprocesseur 6800.

Ces 2 caractéristiques simplifient le dessin des circuits imprimés dans beaucoup d'applications.

4°)Une évaluation grossière de la complexité est:

-environ 6000 transistors,

-dimensions voisines de $4,7 \times 4,7 mm^2 \approx 22 mm^2$.

Le plan de la maquette de simulation logique de ce circuit s'obtiend en rassemblant les plans des sous-ensembles. Cette maquette est très utile pour la validation du fonctionnement, la mise au point de montages et de programmes d'application, et surtout la mise au point des séquences de test des futures puces, travail délicat pour un circuit dont la fonction ne peut guère s'exprimer en quelques mots (voir l'épaisseur du chapitre 2).

8-EXEMPLE D'APPLICATION

8.1-Montage d'application.

8.2-Simulation d'une console TEKTRONIX de la série 4010.

a)Introduction.

b)Principe des consoles TEKTRONIX.

c)Simulation.

8.1-MONTAGE D'APPLICATION

Nous avons réalisé la maquette simulant le fonctionnement du futur circuit intégré. Nous l'avons insérée dans un montage comprenant un microprocesseur avec sa mémoire et ses entrées-sorties (mis au point par J.M.FRAILONG), et une unité d'affichage alphanumérique basée sur le circuit SFF96364 [8] :

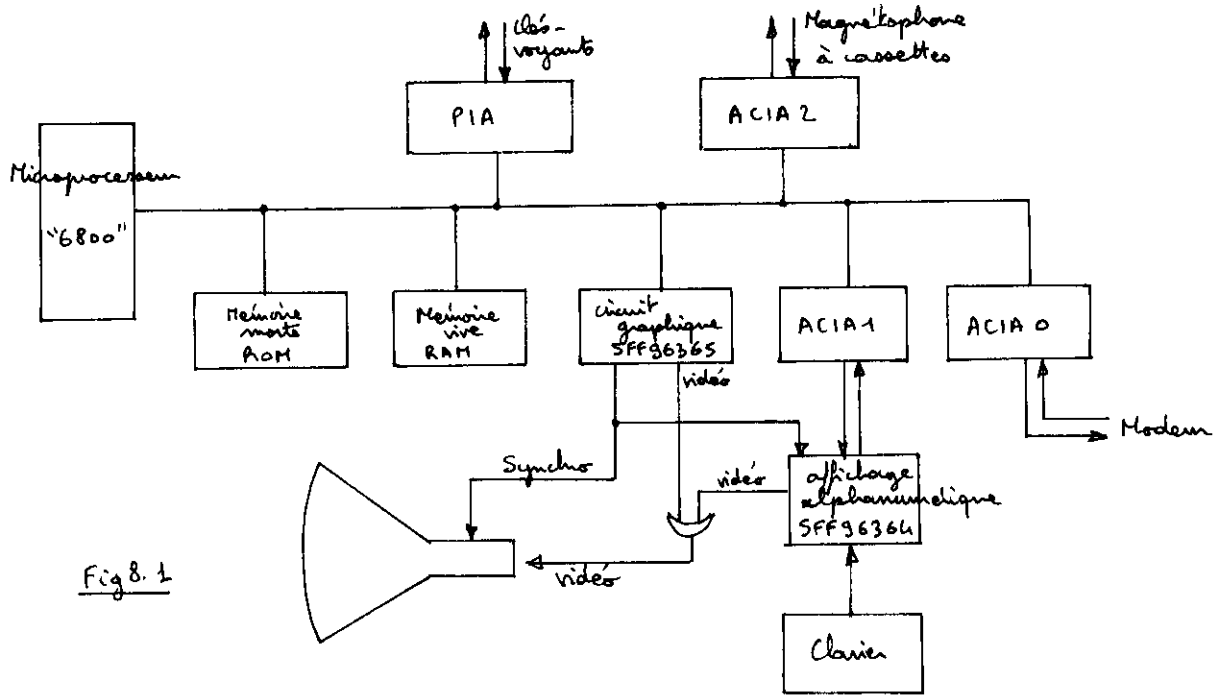


Fig 8.1

Le microprocesseur est un "6800". Le montage permet d'étendre la ROM jusqu'à 8K octets, la RAM jusqu'à 16K octets. Un coupleur parallèle (PIA) permet de brancher un clavier de fonctions avec clés et voyants; 3 coupleurs série asynchrones (ACIA) permettent de coupler la console alphanumérique, un magnétophone à cassettes pour stocker les programmes et une ligne supplémentaire pouvant être couplée à n'importe quel instrument, par exemple un modem téléphonique.

Les 2 affichages graphique et alphanumérique utilisent simultanément le même écran. Ceci est obtenu en synchronisant les 2 systèmes d'affichage et en faisant un OU logique des 2 signaux vidéo.

L'intérêt de la console alphanumérique supplémentaire est de permettre un dialogue avec le moniteur du 6800 lors de la mise au point de la maquette graphique, mais aussi de manipuler simplement l'affichage alphanumérique quand celui-ci n'a pas de rapport direct avec le dessin affiché.

Ce montage nous a permis d'écrire de petits programmes de test des propriétés de la visu graphique, en particulier le dessin de figures en mouvement permettant de tester la vitesse d'affichage.

8.2-SIMULATION D'UNE CONSOLE TEKTRONIX DE LA SERIE 4010

a)Introduction. Il restait à démontrer aux utilisateurs informaticiens qu'une unité d'affichage ainsi conçue pouvait rendre les services qu'ils en attendaient. Mais une part importante du coût de mise en oeuvre d'une telle unité est le coût du logiciel. Or, beaucoup de logiciels graphiques actuels se basent sur l'utilisation du matériel Tektronix.

L'écriture d'un programme sur un microprocesseur permettant de tracer sur l'écran d'un téléviseur le même dessin que celui tracé par une console Tektronix au reçu des mêmes codes permet donc d'utiliser la quasi-totalité des logiciels graphiques existants. Or, l'écriture d'un tel programme est simple: environ 1K octets d'instructions, et une dizaine d'octets de variables.

b)Principe des consoles Tektronix. Ce sont des consoles possédant un écran à mémoire et un générateur de vecteurs analogique. Le spot se trouvant à une adresse, la spécification d'une autre adresse

entraîne le déplacement du spot vers ce nouveau point. Lors de son mouvement, le spot trace le vecteur reliant les 2 points. La précision d'adressage étant 1024x1024 points, une adresse est constituée de 20 bits. La console étant prévue pour se coupler par l'intermédiaire d'une ligne série asynchrone, tous ces 20 bits sont transmis sous forme de 4 caractères, chacun comprenant 5 bits de l'adresse, complétés par des bits d'identification. Dans un autre mode de fonctionnement, la console interprête ces codes en dessinant les caractères associés (ASCII). Le passage d'un mode à l'autre s'effectue à l'aide de 2 caractères de contrôle. Un clavier permet d'émettre des caractères sur la ligne (Fig.8.2)

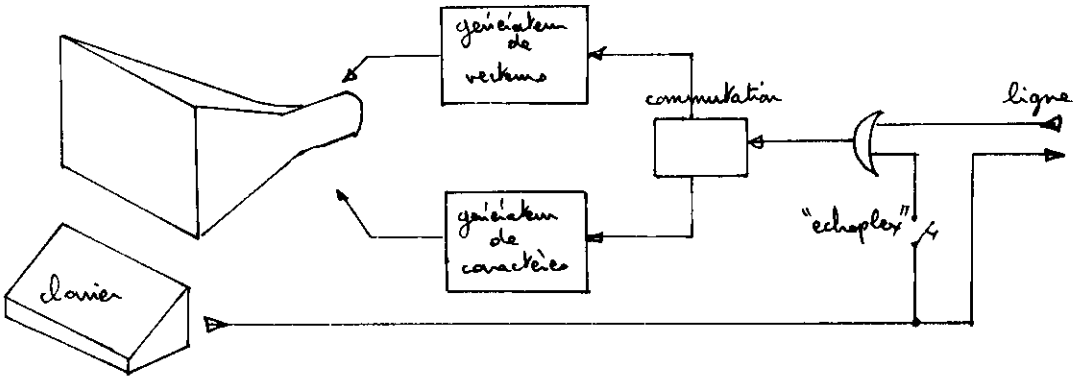


Fig 8.2

Dans la série 4010, il existe plusieurs consoles ayant des performances différentes, mais toutes compatibles entre elles. Un code actif sur une console de bas de gamme a la même action sur une console de haut de gamme; un code actif uniquement sur une console de haut de gamme ne perturbe pas une console de bas de gamme. Il est alors possible de simuler cette gamme en donnant à notre console un ensemble de propriétés que n'ont peut être exactement aucune console de la gamme, mais qui soit compatible avec toutes, modulo cette définition de la compatibilité.

c)Simulation. Notre programme d'émulation du code Tektronix trace des vecteurs sur le circuit graphique, et des caractères sur le circuit SFF96364 à partir de codes caractères échangés sur l'ACIA0. Ce montage a l'inconvénient de dissocier les adresses de spot alphanumérique et graphique , ce qui est une entorse à la compatibilité Tektronix. Nous avons fait des essais en nous couplant à plusieurs centres de calcul et nous avons ainsi pu utiliser des programmes de visualisation de bâtiments en perspective, de surfaces à 3 dimensions, et de masques de circuits intégrés.

Les essais ont été faits avec une définition de 256x256 points, ce qui est un peu faible. Il n'a pas encore été adapté au montage 512x512. Nous comptons le faire prochainement, tout en faisant tracer les caractères alphanumériques dans la mémoire graphique. Nous disposerons alors de 64 lignes de 85 caractères. Nous comptons aussi tester les applications en couleurs sur cette simulation, soit en ajoutant des codes spéciaux, soit en détournant les codes Tektronix de définition des pointillés.

9-CONCLUSION

Le circuit décrit dans ce travail est en cours d'implantation par la société SESCOSEM et verra le jour en 1979. Il permettra de réaliser sur une petite carte de circuit imprimé, accompagnée d'un téléviseur, une console graphique de bonne qualité: définition suffisante, possibilité d'affichage en couleurs, vitesse de tracé élevée permettant l'affichage de figures en mouvement, et ceci pour un coût comparable à celui d'une console alphanumérique. Il n'y a donc plus de raisons d'utiliser des consoles de visualisation sans possibilités graphiques.

Cette réalisation aura été rendue possible par son intégration sur une puce de Silicium (6000 transistors, 22mm^2), qui permet auaujourd'hui une complexité dépassant 20.000 transistors pour un circuit de cette architecture, ce chiffre doublant chaque année.

La part de conception se situant avant l'implantation a occupé une personne pendant environ 2 ans et a nécessité la réalisation de plusieurs maquettes du circuit. Avant d'intégrer une telle fonction, il est nécessaire de repenser complètement l'architecture pour répondre à de nouveaux compromis, tout en profitant de l'expérience accumulée sur le sujet. Nous avons du étudier un générateur de vecteurs et de caractères pour obtenir une vitesse d'inscription maximale compatible avec la technologie, tout en simplifiant au maximum le travail de programmation.

Ceci montre que l'intégration des ordinateurs et des périphériques ne se fait pas automatiquement dès lors que la technologie le permet. Il est remarquable que tous les efforts des constructeurs de circuits intégrés soient presqu'exclusivement axés vers deux types

de circuits: les mémoires et les microprocesseurs. Les mémoires demandent un grand travail de conception électrique, mais le travail d'implantation est relativement faible; les microprocesseurs demandent eux un grand travail de conception logique, mais leur fonction vue de l'extérieur est simple à décrire et s'impose facilement. Or, bien qu'il soit théoriquement possible de construire toutes les machines logiques à base de microprocesseurs et de mémoires, il n'empêche que dans beaucoup de cas, les automates se réalisent plus simples et plus efficaces en utilisant des architectures spécifiques. Le circuit graphique en est un bon exemple. L'automate de visualisation, et surtout les automates de génération des vecteurs et des caractères seraient trop lents et certainement plus gros dans une version microprogrammée.

De tels circuits à fonction spécifique demandent un temps de conception important. Cet inconvénient est aggravé par un temps d'implantation prohibitif. Cette dernière phase demande à être accélérée par l'utilisation d'outils d'implantation (cf [8] et chapitres 0 et 3). En effet, l'investissement de travail que représente l'implantation d'un circuit de 6000 transistors non répétitif est tel qu'il interdit la réalisation de circuits à des fins de recherche, ou de réalisation en petit nombre de machines à structure originale.

Le problème de la réalisation de circuits VLSI se situe bien plus au niveau des méthodes de conception logique et d'implantation de circuits non répétitifs qu'au niveau du contrôle des paramètres électriques ou de la finesse de la gravure. Il faudrait se donner les moyens de manipuler conceptuellement une fonction logique quelconque d'une complexité de 100.000 portes élémentaires en même temps que son implantation. Cette complexité correspond à un millier de

fonctions MSI ou une centaine de fonctions LSI. On peut envisager l'établissement d'un catalogue comprenant un éventail de fonctions de toutes complexités. Un circuit VLSI se concevrait en assemblant de telles fonctions, comme aujourd'hui on les assemble sur un circuit imprimé.

Nous avons proposé un mode de définition hiérarchique des briques, en commençant par décrire les plus petites intéressantes à notre avis et en les agglomérant pour former des briques plus complexes. L'information graphique à manipuler pour décrire l'implantation d'une fonction est donc minimisée, ainsi que le travail de dessin des masques. La validité de cette méthode repose sur un bon choix des briques de base. Nous en avons proposé un, guidé par les nécessités propres au circuit graphique. Nous pensons que ses particularités ne sont pas grandes, et que ce jeu de briques est facilement réutilisable. Néanmoins, une justification ne peut avoir lieu qu'à postériori.

Il est à noter que les plus gros problèmes topologiques d'implantation sont résolus par le choix de ces briques. C'est à dire que, une fois les briques de base choisies, il subsiste très peu de choix dans leur assemblage: les bascules d'un compteur se mettent en file, celles d'un registre aussi. Un additionneur entre ce registre et ce compteur se mettra en "sandwich" entre les deux par exemple, et les sorties de l'additionneur devront alors traverser en parallèle le compteur ou le registre. Ceci suggère qu'une méthode d'implantation semi-automatique peut être constituée seulement d'une bibliothèque bien fournie et d'un éditeur graphique. Si ensuite un "compilateur graphique" s'avère indispensable, sa structure serait beaucoup guidée par les remarques des gens ayant utilisé la bibliothèque.

La mise au point d'un tel catalogue alors que les technologies

de réalisation des fonctions de base évoluent constamment impose de coder ces fonctions de catalogue dans un langage intermédiaire qui s'adapterait à toute technologie par la simple description de l'implantation des portes élémentaires. Cette phase de dessin pourrait alors être manuelle si elle ne doit être faîte que pour les quelques briques de la base de la hiérarchie.

Enfin, on peut reprocher à notre circuit de visualisation graphique de ne permettre qu'une résolution relativement faible, qui ne s'accorde pas avec les performances élevées de la partie écriture (vitesse de tracé des vecteurs par exemple). En particulier, le format 512x512 est limite, puisque l'affichage d'une image en deux trames fait clignoter localement à 25Hz certains éléments tels que les traits horizontaux par exemple. Ce problème peut se résoudre en envisageant la réalisation d'un autre circuit permettant de piloter un moniteur de grande qualité. Ses caractéristiques seraient les suivantes:

-un seul format d'affichage: 1024x1024 points en 50 trames contenant chacune les 1024 lignes. La mémoire d'image serait constituée en Noir et Blanc de 16 boîtiers de 64K bits (annoncés pour l'année 1978),

-bus microprocesseur 12 bits qui permettrait de n'utiliser que 3 bits d'adresse puisque l'adresse du spot tiendrait sur 2 cases mémoire au lieu de 4,

-générateur de vecteurs travaillant sur 12 bits,

-éventuellement un générateur de cercles (voir §1.8),

-l'ensemble tiendrait dans un boîtier 40 broches (Fig.9.1) si les fils de sélection de boîtier sortent non décodés sur 4 broches. La vitesse de fonctionnement du circuit approcherait 5MHz, ce qui n'est pas irréalisable. L'horloge point serait 16 fois plus rapide.

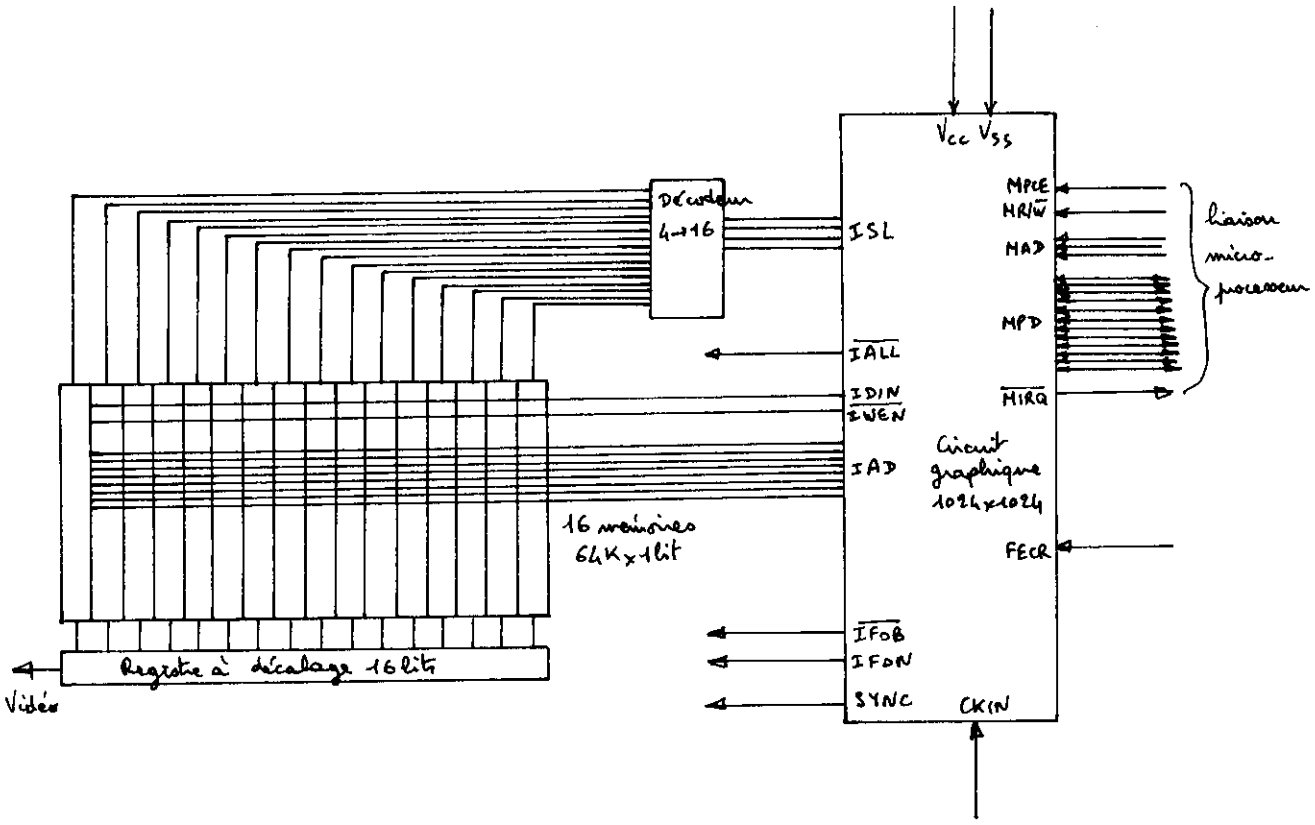

Vcc Vss
Décodeur 4→16
ISL
IALL
IDIN
IWEN
IAD
16 mémoires 64K x 1 bit
Registre à décalage 16 bits
Vidéo
MPCE
HR/W
MAD
MPD
MIRQ
liaison micro-processeur
Circuit graphique 1024x1024
FECR
IFOB
IFON
SYNC
CKIN

Fig 9.1

10-REFERENCES

Le numéro de Septembre 1977 de la revue SCIENTIFIC AMERICAN intitulé "Microelectronics",(Vol.237 n°3) constitue une excellente introduction aux circuits intégrés, en particulier les 3 articles suivants:

[1] R. N. Noyce, "Microelectronics", p.62-69.

[3] I. E. Sutherland et C. A. Mead, "Microelectronics and Computer Science", p.210-228

[5] A. C. Kay, "Microelectronics and the Personal Computer", p.230-244.

[2] I.B.M. Journal of Research and Development, Vol.21 n°6, Nov.1977, p.498-521.

[4] J. Mc Carthy, "The Home Information Terminal", preprint 1977.

[6] P. Morvan et M. Lucas, "Images et Ordinateur, introduction à l'infographie interactive", Larousse Université 1976.

[7] S. J. Op Het Veld, "Microprocessor controlled Video Games", Euromicro 1977.

[8] J. Gastinel, "Conception et Intégration d'un Terminal Alphanumérique", Thèse de 3ème cycle, Institut de Programmation, Paris 1977.

[9] Notices des circuits de visualisation VRAMs de MATROX.

[10] Présentation de la console Hewlett-Packard 2648A.

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

[12] Notice technique de la mémoire Mostek 4116.

[13] Intel Data Catalog 1977, mémoire 2116, p.2-96.

[14] W. N. Carr and J. P. Mize, "MOS/LSI Design and Application", Mc Graw Hill 1972.

[15] J. T. Quatse and R. A. Keir, "A Parallel Accumulator for a General-Purpose Computer", I.E.E.E. Transactions on electronic computers, Vol.EC-16, n°2, April 1967.

[16] J. E. Bresenham, "Algorithm for Computer Control of a digital Plotter", IBM Systems J-4, 1965, p.25-30.

[17] B. K. P. Horn, "Circle Generators for display devices", Computer Graphics and Image Processing 5, p.280-288, 1976.

[18] A. Bernardy, "Une Conception Nouvelle des Equipements Périphériques appliquée à la Visualisation Graphique", Thèse de 3ème cycle, Institut de Programmation, Paris, Février 1974.

[19] "The TTL Data Book for Design Engineers", catalogue Texas Instruments, 2ème édition (Vol.LCC4112).

[20] P. Matherat, "Réalisation d'une Unité de Visualisation Graphique sur Ecran de Télévision", rapport de stage de D.E.A., Institut de Programmation, Juin 1976.

[21] R. M. M. Oberman, "A Flexible Rate Multiplier Circuit with Uniform Pulse Distribution Outputs", I.E.E.E. Transactions on electronic computers, August 1972, Vol.C-21, p.896-899.

[22] Notice utilisateurs des terminaux graphiques Tektronix de la série 4010.

11-ANNEXE: BROCHAGE DU CIRCUIT

11.1-Diagramme fonctionnel.

11.2-Brochage.

11.3-Représentation symbolique.

11.4-Description des broches.

11.1-DIAGRAMME FONCTIONNEL

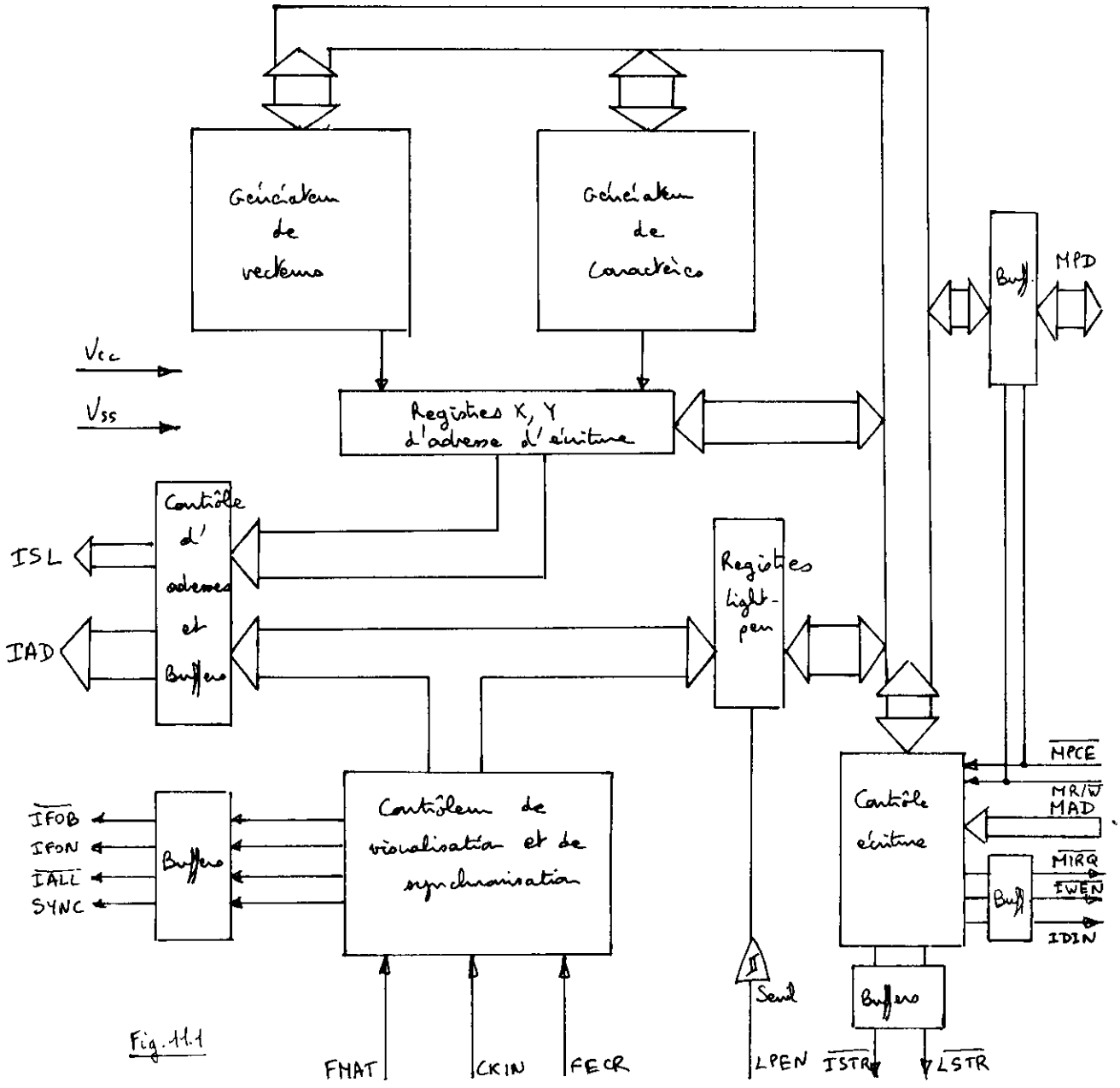


Fig. 11.1

11.2-BROCHAGE

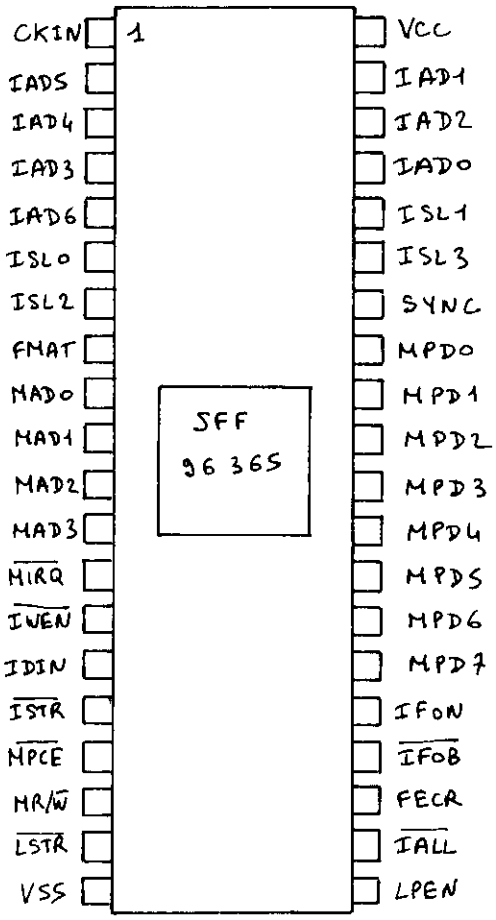


Fig 11.2

11.3-REPRESENTATION SYMBOLIQUE

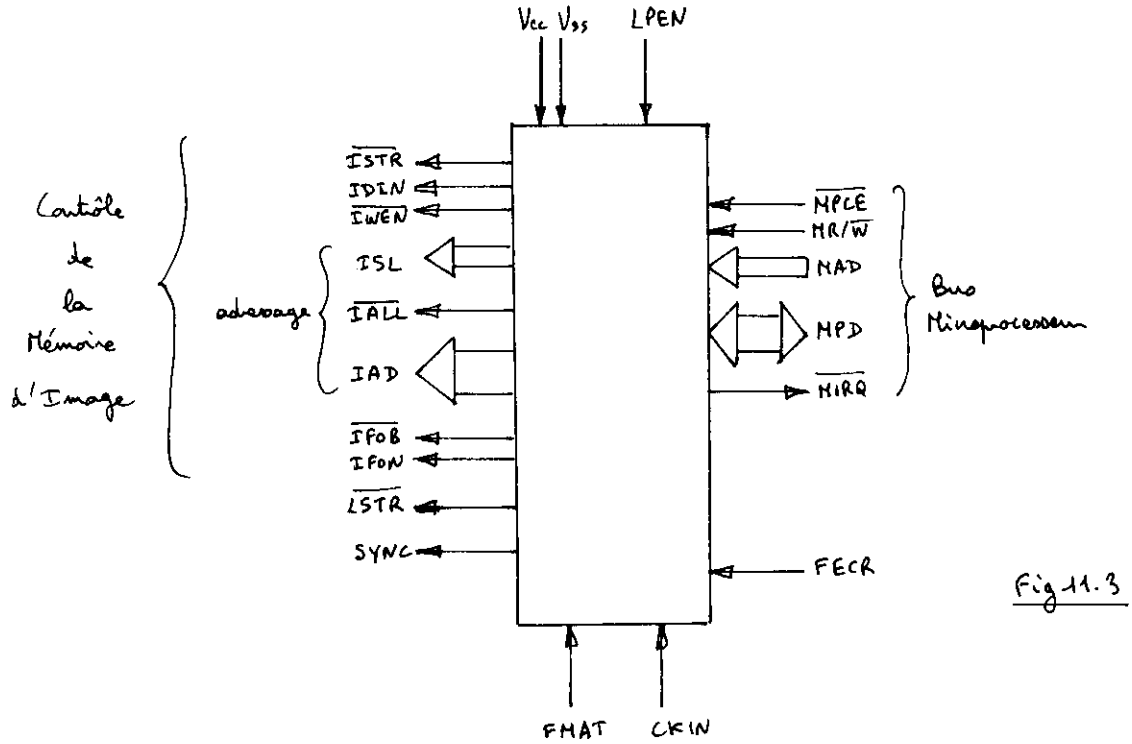


Fig 11.3

11.4-DESCRIPTION DES BROCHES

Nom du signal	E/S	Fonction
Vcc		
Vss		Broches d'alimentation du boîtier.
CKIN	E	Signal d'horloge du boîtier. Tous les compteurs et registres internes sont modifiés sur un front descendant de ce signal. Si ce signal est bas, c'est la partie basse des adresses de la mémoire d'image qui est présente sur IADi, et vice-versa. Pour assurer une bonne synchronisation du récepteur TV, il est recommandé d'appliquer sur cette broche un signal TTL de fréquence 1,750MHz si FMAT est à Vcc, 1,7472MHz si FMAT est à Vss.
FMAT	E	Format. Doit être connecté à Vcc pour une résolution verticale de 512 lignes, et à Vss pour 256 lignes ou moins. Cette entrée change la forme des signaux de synchronisation, la répartition des adresses sur IADi, et la fonction des sorties ISL.
FECR	E	Forçage en écriture. Si cette entrée est au niveau haut, La mémoire n'est plus rafraîchie, toutes les périodes d'horloge peuvent être des périodes d'écriture.
SYNC	S	Signal de synchronisation du récepteur TV. Celui-ci doit être commuté sur un balayage 625 lignes. Si FMAT est connecté à Vcc, ce signal est aux normes françaises. Si FMAT est connecté à Vss, les trames ne sont plus entrelacées, et comprennent toutes 312 lignes.

Signaux de contrôle de la mémoire d'image:

IADO-6	S	Adresses de la mémoire d'image.
ISLO-3	S	Sélection des boîtiers mémoire d'image. 1°)Fonctionnement avec FMAT à Vss: ISLi portent directement les signaux $\overline{RAS}$ (dérivés de CKIN) pour l'utilisation des mémoires 16Kx1bits ou 4K x1bits, 16 broches (voir schémas d'application). 2°)Fonctionnement avec FMAT à Vcc (512x512): ISLi portent le numéro codé sur 4 bits du boîtier sélecté. La sélection commune de 8 boîtiers pour la lecture et le rafraîchissement doit être assurée par l'action conjuguée de $\overline{IALL}$ et ISL3.
$\overline{IALL}$	S	Signal bas lors de la lecture et du rafraîchissement.
$\overline{IWEN}$	S	Signal $\overline{WE}$ des mémoires d'image.
IDIN	S	Signal de forçage (actif haut) de la donnée d'entrée des mémoires à l'état éteint
$\overline{ISTR}$	S	Créneau reproduisant le $\overline{MPCE}$ d'accès à l'adresse H'A'.
$\overline{IFOB}$	S	Forçage à blanc. Signal destiné à forcer au niveau bas la sortie des mémoires pour forcer le signal vidéo à l'état blanc lors de l'utilisation du light-pen, ainsi qu'en début de trame (actif bas).
IFON	S	Forçage à noir. Signal d'effacement vidéo destiné forcer à l'état éteint le signal vidéo sur les bords de l'image (actif haut).
$\overline{LSTR}$	S	Créneau négatif permettant d'enregistrer la sortie des mémoires après l'envoi du code de commande H'F'.

Signaux de dialogue avec le microprocesseur:

MPDO-7	E/S	Bus bidirectionnel de données. Si $\overline{\text{MPCE}}$ est haut, ces broches sont à l'état hautes impédance.
MADO-3	E	Bus d'adresse. Permet de sélectionner le registre concerné par une lecture ou une écriture.
MR/$\overline{\text{W}}$	E	Si ce signal est bas, il y aura enregistrement du bus de données dans le registre concerné sur le front arrière du $\overline{\text{MPCE}}$. Si ce signal est haut, le contenu du registre désigné sera présenté sur le bus de données lors de $\overline{\text{MPCE}}$.
$\overline{\text{MPCE}}$	E	Signal temporel d'écriture ou de lecture (actif bas).
$\overline{\text{MIRQ}}$	S	Requête d'interruption. Sortie à collecteur ouvert.

Light-pen:

LPEN	E	Signal du light-pen, actif sur front montant.